450 – Linux Essentials

www.4linux.com.br

Sumário

Capítulo 1

Introdução ao infinito

1.1. Objetivos

- Descobrir o que é GNU/Linux;
- Entender a filosofia do Software Livre;
- Conhecer um pouco da história;
- Entender o que são Distribuições;

1.2. Introdução

Utilizar um sistema GNU/Linux é muito mais do que optar por uma solução isenta de custos de licença. É usufruir uma filosofia que antecedeu o software proprietário, e que permitiu, por exemplo, que a internet crescesse de forma aberta como a conhecemos hoje. Como usuário de software livre, precisamos compreender um pouco mais sobre essa ideologia e como ela promoveu o surgimento das várias distribuições.

O sistema GNU/Linux é frequentemente chamado apenas pelo seu segundo nome, Linux. Entretanto, essa designação não faz justiça a todos os desenvolvedores que vêm desenvolvendo o sistema.

GNU, que é um acrônimo recursivo de GNU's not Unix, é um grupo que foi fundado em 1984 por seu idealizador, Richard Stallman, com o intuito de criar um sistema operacional ``Unix-Like''. Sendo assim, diversos softwares passaram a ser criados e mantidos pela comunidade que se formara, entretanto, havia um pedaço de código essencial que ainda não tinha sido criado: o kernel.

Em 1991, um jovem finlandês chamado Linus Torvalds disponibilizou para o mundo a primeira versão do Linux, um kernel ``Unix-Like''. A partir desse ponto, foi possível unir o kernel - Linux - com os softwares GNU, originando o que chamamos de GNU/Linux.

O mundo GNU/Linux não é apenas um conjunto de programas mas também uma filosofia de mundo livre e colaborativo, no qual as pessoas podem utilizar esses softwares livremente e, acima de tudo, aprender com eles, uma vez que seu código fonte deve ser disponível a todos que queiram melhorá-lo ou apenas aprender com ele. Para que esse mundo continue livre, Richard Stallman fundou a FSF - Free Software Foundation, que mantém a licença chamada GNU GPL - GNU General Public License.

- liberdade 0 - liberdade para rodar o programa para quaisquer propósitos;
- liberdade 1 - liberdade para estudar como o programa trabalha e adaptá-lo às suas necessidades. Ter acesso ao código fonte é essencial para isso.
- liberdade 2 - liberdade de redistribuir cópias de forma que você possa ajudar outras pessoas.
- liberdade 3 - liberdade para melhorar o programa e disponibilizar as melhorias para o público, de forma que toda a comunidade possa se beneficiar disso. Ter acesso ao código fonte é essencial também para isso.

Após a criação dessa licença, várias outras licenças, usando a filosofia de copyleft, foram criadas com o objetivo de defender a liberdade do conhecimento, informação e do código aberto. Abaixo podemos ver alguns exemplos:

- GFDL - GNU Free Documentation Licence (www.gnu.org/copyleft/fdl.html);
- OPL - Open Publication License (http://www.opencontent.org/openpub/);
- CC - Creative Commons (http://creativecommons.org/about/licenses);
- BSD - Berkeley Software Distribution (http://www.freebsd.org/copyright/license.html);
- SPL - Sun Public Licence (http://java.sun.com/spl.html);

Atualmente a GPL está disponível em três versões, GPLv1, GPLv2 e GPLv3. Fique por dentro de suas diferenças em: http://www.gnu.org/licenses/gpl.html

Para mais informações a respeito do kernel - Linux - podem ser obtidas no site oficial de seus mantenedores: http://www.kernel.org

A respeito do GNU e da FSF podem ser obtidas nos sites

- http://www.gnu.org
- http://www.fsf.org.

1.3. Distribuições GNU/Linux

Você já deve ter ouvido falar em Debian, RedHat, Slackware, SuSe, Conectiva, Mandrake, Ubuntu dentre outras. Mas, o que realmente é isso? O que são todos esses nomes? Todos esses nomes são o que chamamos de distribuições GNU/Linux. Uma distribuição nada mais é do que o kernel, Linux, softwares GNU e outros aplicativos que são desenvolvidos por outras comunidades ou grupos.

Mas, por que tantas distribuições? Justamente porque se você não se identifica com nenhuma delas, você é livre para fazer a sua própria. Ou seja, em 1993, um rapaz chamado Patrick Volkerding, juntou o kernel e vários outros aplicativos em uma distribuição chamada Slackware, que foi a primeira a ser distribuída em CD. A partir desse ponto, foram surgindo diversas outras distribuições que de alguma forma diferiam da filosofia do Slackware: como Debian ou RedHat,

por exemplo.
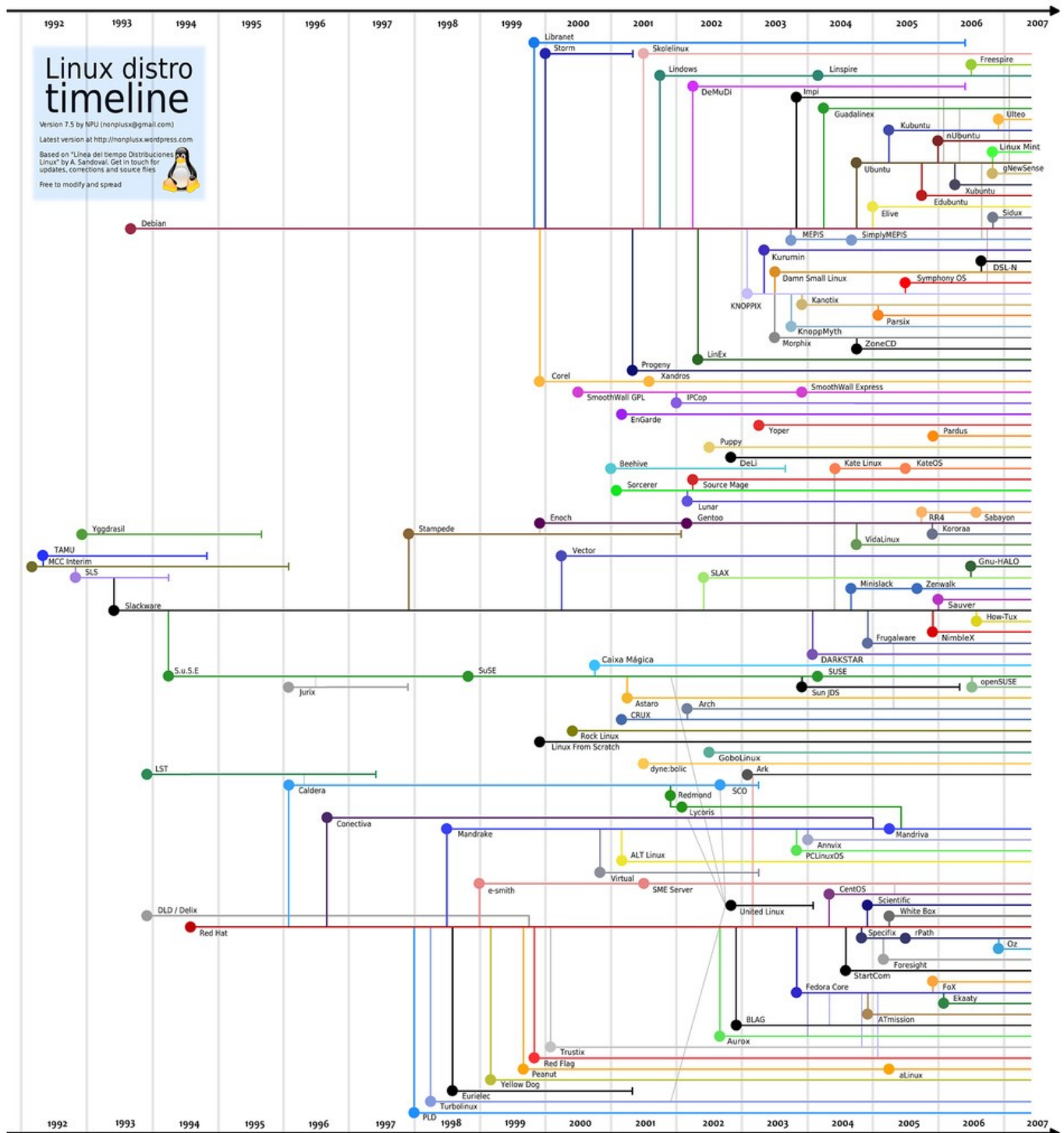
Ilustração 1: Linha de tempo GNU/Linux

Atualmente existem centenas de distribuições, algumas mais famosas que outras. Em sua maioria, as distribuições GNU/Linux são mantidas por grandes comunidades de colaboradores, entretanto, há outras que são mantidas por empresas. Dessa forma, podemos dividir as distros, abreviação bastante utilizada na comunidade e que se refere às distribuições, em duas categorias básicas:

- Livres
- Corporativas

Distribuições Livres - mantidas por comunidades de colaboradores sem fins lucrativos. Exemplos são: Debian, Ubuntu, Slackware, Gentoo, Knoppix e CentOS, entre outras.

Distribuições Corporativas - mantidas por empresas que vendem o suporte ao seu sistema. Exemplos são: RedHat, SuSe e Mandriva.

Neste ponto vale ressaltar o fato de que o produto vendido pelas empresas que comercializam sistemas GNU/Linux, são na verdade, os serviços relacionados ao sistema vendido, como suporte técnico, garantias e treinamentos, ou seja, o conhecimento do sistema. O fato de o produto não ser mais o software, mas sim o serviço, é devido à Licença GPL que garante as já citadas quatro liberdades básicas. Com isso, por mais que uma empresa queira fazer o seu próprio sistema GNU/Linux, enquanto ela estiver utilizando softwares registrados com GPL, serão obrigadas a distribuir o código fonte gratuitamente.

Dentro do conjunto de Distribuições Livres, podemos dividi-las novamente em duas outras categorias:

- Convencionais
- Live

Distribuições Convencionais- são distribuídas da forma tradicional, ou seja, uma ou mais mídias que são utilizadas para instalar o sistema no disco rígido;

Distribuições Live - são distribuídas em mídias com o intuito de rodarem a partir delas, sem a necessidade de instalar no HD. As distribuições Live ficaram famosas pois têm a intenção de fornecer um sistema GNU/Linux totalmente funcional, de forma fácil e sem a necessidade de o instalar na máquina. O fator que favoreceu essa abordagem é que em uma distribuição Live praticamente todos os componentes já vêm configurados, funcionando e com interfaces agradáveis aos usuários finais. Exemplos desse tipo de distribuição são o Knoppix, do qual se originaram diversas outras como Kurumin ou Kalango, que são versões brasileiras do Knoppix, e o Ubuntu, bastante difundido atualmente.

Ainda para entender um pouco mais das distribuições, é necessário lembrar de mais duas características:

- From scratch
- Provenientes (Baseadas)

Distribuições From Scratch - São desenvolvidas do zero, ou seja, utiliza um kernel linux, alguns programas GNU e a grande maioria das suas particularidades é desenvolvida específicamete para ela. Exemplos:

- Debian ;
- RedHat;
- Gentoo;
- Slackware;
- entre outras;

Distribuições Provenientes (Baseadas) - Aproveitam ferramentas e bases já desenvolvidas por outras distribuições. Distribuições baseadas usam distribuições from scratch para alcançar seus objetivos mais rápido, dando maior atenção para ao propósito da distribuição. Exemplos: Ubuntu, DreamLinux, Kubuntu, Kurumin, Slax, BrDesktop entre muitas outras.

1.4. Características de Algumas Distribuições

Será mostrado a seguir a característica de algumas distribuições. Você pode encontrar uma lista das distribuições existentes, bem como das estatísticas de downloads, no site:

- *http://distrowatch.com*

1.4.1. RedHat

- **Tipo: corporativa;**
- **Descrição:** primeira distribuição corporativa a ser criada. Muito utilizada nas empresas por oferecer suporte técnico e ter seu sistema compatível com as diversas tecnologias disponíveis;
- **Interface padrão:** GNOME;
- **Sistema de pacote:** RPM - RedHat Package Manager;
- **Site oficial:** http://www.redhat.com

1.4.2. SuSe_

- **Tipo:** corporativa;
- **Descrição:** Comprada pela Novell em 2003, é a principal concorrente da RedHat, atuando no meio corporativo tanto em servidores quanto em desktops. Assim como a RedHat, possui parcerias com diversas empresas, a fim de manter seu sistema compatível com produtos de terceiros;
- **Interface padrão:** GNOME;
- **Sistema de pacote:** baseado em RPM, mas não segue o formato da RedHat à risca, tendo implementado algumas variações;
- **Site oficial:** http://www.novell.com/linux

1.4.3. Mandriva_

- **Distribuição:** corporativa;
- **Descrição:** originada da fusão da Mandrake e Conectiva, especializada em serviços e projetos embarcados;
- **Interface padrão:** KDE;
- **Sistema de pacote:** RPM;
- **Site oficial:** http://www.mandriva.com

1.4.4. Slackware_

- **Distribuição:** livre;
- **Descrição:** primeira distribuição GNU/Linux a ser distribuída em CD, é considerada como sendo a primeira distribuição. Organizada por seu criador Patrick Volkerding, caracteriza-se por sua filosofia de ser a distribuição mais ``Unix-Like'' do mundo GNU/Linux.
- **Interface padrão:** KDE;
- **Sistema de pacote:** tgz;
- **Site oficial:** http://www.slackware.com

1.4.5. Debian_

- **Distribuição**: livre;
- **Descrição:** criada com o intuito de prover um sistema operacional totalmente livre e gratuito, foi uma das primeiras distribuições GNU/Linux a serem criadas. Atualmente é uma das maiores distribuições e a que mais gerou distribuições derivadas. Por ser uma referência em sistemas GNU/Linux, é a distribuição mais utilizada em órgãos públicos e governos;
- **Interface padrão:** GNOME;
- **Sistema de pacote:** DEB - Debian Package;
- **Site oficial:** http://www.debian.org

1.4.6. Ubuntu

- **Distribuição**: livre (convencional e Live);
- **Descrição:** com seu slogan Linux for Human Beings - é voltada para o usuário final, apesar de ter versão para servidores. Patrocinada pelo milionário Mark Shuttleworth é, atualmente, a maior distribuição em número de downloads.
- **Interface padrão:** GNOME ou KDE (para Kubuntu);
- **Sistema de pacote:** DEB - Debian Package;
- **Site oficial:** http://www.ubuntu.com

1.4.7. Fedora

- **Distribuição**: livre;
- **Descrição:** mantida pela RedHat, serve de teste para o carro chefe da empresa, o RedHat Enterprise.
- **Interface padrão:** GNOME;
- **Sistema de pacote:** RPM - RedHat Package Manager;
- **Site oficial:** http://fedora.redhat.com

1.4.8. OpenSuSe

- **Distribuição:** livre;
- **Descrição:** patrocinada pela Novell, baseia-se no SuSe Linux.
- **Interface padrão:** GNOME ou KDE;
- **Sistema de pacote:** RPM;
- **Site oficial:** http://en.opensuse.org

1.4.9. Knoppix

- **Distribuição**: livre (Live);
- **Descrição:** distribuição Live que popularizou distribuições do gênero, devido à sua ferramenta de remasterização que facilitou o processo de gerar novas distribuições;
- **Interface padrão:** KDE;
- **Sistema de pacote:** DEB;
- **Site oficial:** http://www.knoppix.org

1.4.10. Gentoo

- **Distribuição:** livre (Live);
- **Descrição:** Todos os programas são compilados na própria máquina. As principais vantagens são a performance e a personalização conforme as necessidades do usuário. A principal desvantagem é o trabalho e tempo necessários a sua instalação.

- **Interface padrão:** A escolha do usuário;
- **Sistema de pacote:** Emerge, código fonte;
- **Site oficial:** http://www.gentoo.org

Exercícios Teóricos

1) Quais são as liberdades básicas idealizadas por Richard Stallman?

____
____
____
____

2) O que é Linux e quem é o seu criador?

____
____
____
____

3) O que é uma distribuição GNU/Linux?

____
____
____
____

4) O que é GPL?

____
____
____
____

5) Qual é a finalidade das licenças que defendem o copyleft?

____
____
____
____

6) Software livre é o mesmo que software grátis?

____

____

____

____

7) Qual foi a primeira distribuição disponibilizada em CD?

____

____

____

____

8) Qual a diferença entre uma distro live e convencional?

____

____

____

____

9) Como funciona o processo de desenvolvimento de uma distribuição GNU/Linux?

____

____

____

____

Capítulo 2

Primeiros Passos

2.1. Objetivos

- Entender a estrutura do sistema operacional;
- Descobrir as funcionalidades do Shell;
- Executar os primeiros comandos no sistema;

2.2. Entendendo a estrutura do Linux

Como podemos ver na figura abaixo, o sistema operacional GNU/Linux pode ser dividido em algumas layers:
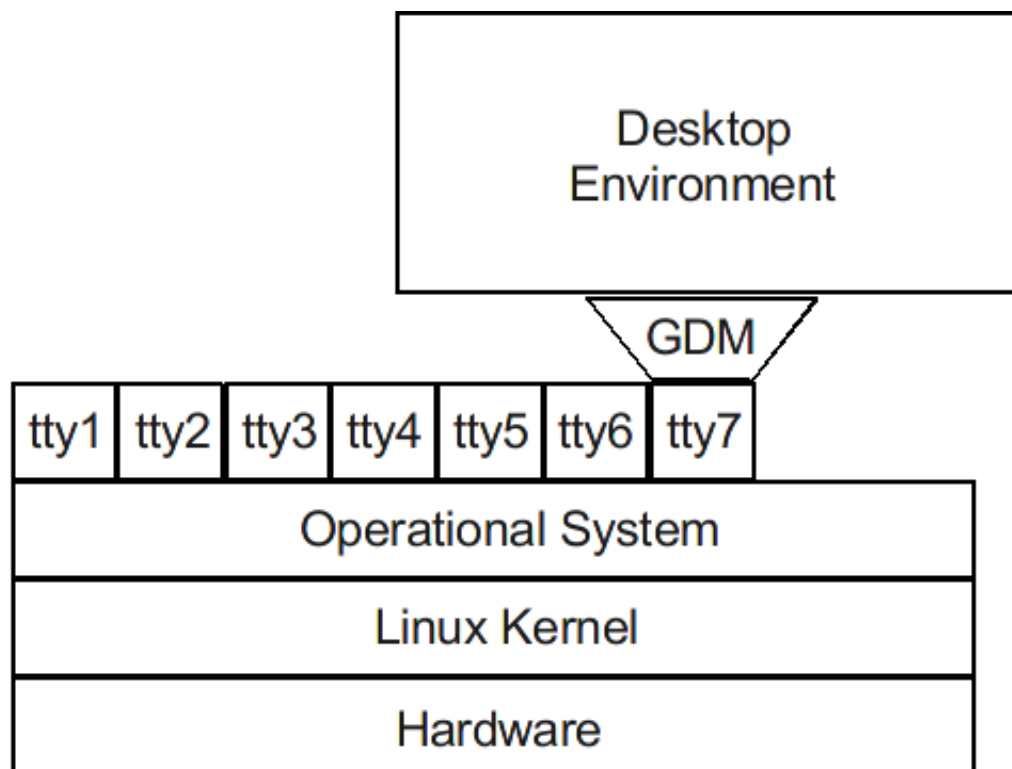
Ilustração 2: Estrutura do Sistema

Para entendermos melhor vamos descrever cada uma delas:

- **Hardware -** Dispositivos que estão disponíveis para o uso do sistema, tais como cd-rom, placa de rede, controladora scsi entre outros;
- **Kernel -** O núcleo do sistema operacional, essa layer é quem faz todas as interações com o hardware da máquina, interpretando todas as requisições das layers acima;
- **Sistema Operacional -** Essa layer tem como função auxiliar e abrigar todos os aplicativos das layers superiores. Segundo Linux torvalds esse layer não deve ser notada por um usuário leigo final;
- **ttyN -** Terminais Virtuais aonde são executados comandos e seta-das configurações. As tty's interpretam os comandos dados por um humano e converte os mesmos para uma linguagem que a máquina entenda;
- **DM -** A Layer de Display Manager é responsável por gerenciar os logins na interface gráfica e escolher o tipo de ambiente gráfico que deve ser executado;
- **Desktop Environment -** Mais conhecido como gerenciador de janela, é responsável por abrigar todos os programas que necessitam um gerenciador de janelas, e por deixar o ambiente mais agradável.

2.3. Introdução ao Shelll

No Mundo GNU/Linux, utilizamos o shell, que funciona como interpretador de comandos. Inicialmente devemos saber como usá-lo. O shell é a interface entre o usuário e o kernel do sistema e por meio dele, podemos digitar os comandos. O shell padrão do GNU/Linux é o bash. Existem também outros shells, como, por exemplo, csh, tcsh, ksh e zsh.

O kernel é a parte mais próxima do hardware do computador. É o núcleo do Sistema Operacional. Se seu GNU/Linux estiver com problemas, não chute seu computador, a culpa não é dele.

O local onde o comando será digitado é marcado por um traço piscante na tela, chamado de cursor. Tanto em shells texto como nos shells gráficos é necessário o uso do cursor para sabermos onde devemos iniciar a digitação de textos e nos orientarmos quanto à posição na tela.

Popularmente conhecido como linha de comandos, o shell interpreta o usuário que irá efetuar uma ação de duas maneiras, são elas:

- **Super usuário**, popularmente conhecido como **root**. Não se engane, root não é de raiz, da língua inglesa. O usuário root é o administrador do sistema, e seu diretório (pasta) padrão é o /root, diferentemente dos demais usuários que ficam dentro de /home. No próximo capítulo falaremos mais sobre a estrutura de diretórios do GNU/Linux. O shell de um usuário root é diferente de um usuário comum. Antes do cursor, ele é identificado com ``#'' (jogo-da-velha).
- **Usuário comum**, qualquer usuário do sistema que não seja root e não tenha poderes administrativos no sistema. Como já havíamos dito anteriormente, o diretório padrão para os usuários é o /home. Antes do cursor, o shell de um usuário comum é identificado com ``$'' (cifrão).

Existem muitas funcionalidades no shell, uma delas é retornar comandos que já foram digitados anteriormente. Para fazer isso é só pressionar as teclas seta para cima e seta para baixo, caso queira retornar.

Outra funcionalidade também muito utilizada, serve para visualizarmos a nossa tela de modo que possamos ir para cima ou para baixo, parecido com o scroll. Para rolarmos a tela para cima, seguramos o Shift e pressionamos o Page Up. Para

rolarmos a tela para baixo, seguramos o Shift e pressionamos o Page Down. Isto é útil para ver textos que rolaram rapidamente para cima.

Existem duas formas de executar comandos como o administrador do sistema, logando como root e usando os comandos su e sudo.

- **su -** Para usar o comando su é necessário ter o password do administrador, uma vez executado é possível executar qualquer comado como administrador do sistema.
- **sudo -** Já para a utilização do comando sudo é necessário ter a senha do usuário corrente. Esse usuário também precisa estar presente na lista de usuários do sudo, que apenas o administrador tem acesso através do comando:

```
# visudo
```

A forma de se utilizar o comando sudo é diferente, já que ele da permissões de execução para apenas um comando.

Utilização com comando su:

```
$ su <user>
```

Utilização com comando sudo:

```
$ sudo <comando >
```

2.4. Terminal Virtual

Terminal (ou console) é o teclado e a tela conectados em seu computador. O GNU/Linux faz uso de sua característica multi-usuário, ou seja, suporta vários usuários, usando os ``terminais virtuais''. Um terminal virtual é uma segunda seção de trabalho completamente independente de outras e que pode ser acessado no computador local ou remotamente, utilizando os programas telnet, rsh, rlogin, rdesktop, vnc, ssh, etc. Nos dias de hoje, o acesso remoto é muito importante. A qualquer distância que esteja o cliente, é possível atendê-lo.

No GNU/Linux é possível, em modo texto, acessar outros terminais virtuais, segurando a tecla ALT e pressionando F1 até F6. Cada tecla tem função correspondente a um número de terminal do 1 ao 6, isso é por default, e pode ser mudado (o sétimo, por default, é usado pelo ambiente gráfico X-Window-System).

O GNU/Linux possui mais de 63 terminais virtuais, mas deles, apenas 6 estão disponíveis, inicialmente por motivos de economia de memória RAM. Se você estiver usando o modo gráfico, deve segurar Ctrl+Alt enquanto pressiona uma tecla de atalho de F1 a F6.

Um exemplo prático: se você estiver utilizando o sistema no terminal 1, pressione Ctrl+Alt + F2, e veja na primeira linha nome e versão do sistema operacional, nome da máquina e o terminal que você está. Você pode utilizar quantos terminais quiser, do F1 ao F6 (inclusive utilizando o X) e pode ficar ``saltando'' de terminal para terminal.

2.5. Logon

Logon é a entrada do usuário, root ou comum, onde deve ser digitado seu nome de usuário, e logo depois sua senha. Caso você digite algo de forma errada, irá aparecer uma mensagem de erro e você não será logado no sistema.

2.6. Histórico de comandos

O terminal do linux permite que você guarde 500 comandos por padrão, assim não precisa redigitar o comando quando precisar dele denovo.

```
$ history
```

2.7. Logout

Logout é a saída do sistema. Ela é feita pelos comandos

```
$ logout
$ exit
$ <CTRL>+D
```

ou quando o sistema é reiniciado ou desligado.

2.8. Desligando o Computador

Para desligar o computador, primeiro digite um dos comandos abaixo:(como root):

```
# shutdown -h now
# halt
# poweroff
```

A palavra halt vem do comando em assembly chamado HTL, que quer dizer ``parada de processamento''. Assim, o GNU/Linux finalizará os programas e gravará os dados em seu disco rígido. Quando for mostrada a mensagem ``power down'', pressione o botão POWER em seu gabinete para desligar a alimentação de energia do computador. NUNCA desligue o computador diretamente sem utilizar o comando shutdown, halt ou poweroff, pois podem ocorrer perdas de dados ou falhas no sistema de arquivos de seu disco rígido, devido a programas abertos e dados ainda não gravados no disco. Os comandos halt e poweroff disparam uma série de procedimentos, como encerramento de serviços e desligamento de sistemas de arquivos, que são executados antes da máquina ser desligada.

Salve seus trabalhos para não correr riscos de perdê-los durante o desligamento do computador. Tenha um Nobreak.

O comando shutdown tem a seguinte sintaxe:

```
# shutdown <ação> <tempo>
```

Onde:

- **ação** - o que você quer fazer, As opções são:
  - **-h** para desligar
  - **-r** para reiniciar.
- **tempo** - tempo em minutos que você deseja para começar a executar a ação.

Exemplo:

Desligar agora:

```
# shutdown -h now
```

Desligar daqui a 12 minutos:

```
# shutdown -h 12
```

2.9. Reiniciando o Computador

Reiniciar quer dizer ``Iniciar novamente o sistema''. Não é recomendável desligar e ligar constantemente o Computador pelo botão ON/OFF ou RESET. Por isso, existem recursos para reiniciar o sistema sem desligar o computador. No GNU/Linux você pode usar o comando reboot, shutdown -r now e também pressionar simultaneamente as teclas para reiniciar de forma segura.

Observações:

- Salve seus trabalhos.
- Utilize comandos e não o dedo.

- Prefira o método de reinicialização explicado acima e use o botão reset somente em último caso.

Reiniciar agora:

```
# shutdown -r now
```

Reiniciar daqui a 5 minutos:

```
# shutdown -r 5
```

2.10. Prática Dirigida

A seguir, vamos testar algumas funcionalidades da linha de comandos (não é necessário se preocupar em decorá-los, com o passar do tempo, pegamos um pouco mais de prática):

- Pressione a tecla **Back Space** para apagar um caractere à esquerda do cursor;
- Pressione a tecla **Delete** para apagar o caractere acima do cursor;
- Pressione a tecla **Home** para ir ao começo da linha de comando;
- Pressione a tecla **End** para ir ao final da linha de comando;
- Pressione as teclas **Ctrl + A** para mover o cursor para o início da linha de comandos;
- Pressione as teclas **Ctrl + E** para mover o cursor para o fim da linha de comandos;
- Pressione as teclas **Ctrl + U** para apagar o que estiver à esquerda do cursor. O conteúdo apagado é copiado para uso com **Ctrl + y;**
- Pressione as teclas **Ctrl + K** para apagar o que estiver à direita do cursor. O conteúdo apagado é copiado para uso com **Ctrl + y;**
- Pressione as teclas **Ctrl + L** para limpar a tela e manter a linha de comando na primeira linha. Mas se você der um **Shift + Page Up** você ainda consegue enxergar o conteúdo. O **Ctrl + L** funciona igual ao comando ``clear'', que tem a mesma função;
- Pressione as teclas **Ctrl + C** para abrir uma nova linha de comando, na posição atual do cursor;

- Pressione as teclas **Ctrl + D** para sair do shell. Este é equivalente ao comando ``**exit**'';
- Pressione as teclas **Ctrl + R** para procurar ``**x**'' letra relacionada ao último comando digitado que tinha ``**x**'' letra como conteúdo do comando;

Coloque o computador para desligar em 1 minuto:

```
# shutdow -h 1
```

Ligue o micro:

Faça o login:

Usuário: aluno

Senha: 123456

Coloque o computador para reiniciar em 1 minuto:

```
# shutdown -r 1
```

Deu erro??? Por que???

Eleve seus poderes no sistema, vire root através do comando:

```
# su
password: 123456
```

E agora, você consegue reiniciar a máquina:

```
# shutdown -r now
```

2.11. Exercício Teórico

1) Você precisa desligar a máquina, mas alguém está usando sua impressora e isso o impede de dar o boot imediatamente. Sabendo que o trabalho da impressão termina no máximo em 5 minutos, qual comando você usaria para desligar a máquina daqui a 10 minutos?

2) Você precisa enviar um aviso para seus 150 usuários logados, mas sem desligar ou reiniciar a máquina. Como você faria?

3) É correto afirmar que quase todos os programas gráficos são provenientes de um comando?

4) Cite qual é a principal função do terminal de comandos?

5) Qual é a quantidade de terminais que estão disponíveis no sistema operacional Debian GNU/Linux?

Capítulo 3

Sistema de Arquivos e Diretórios

3.1. Objetivos

- Entender o que é FHS;
- Conhecer a estrutura de diretórios do sistema;
- Descobrir alguns diretórios e suas determinadas finalidades;

3.2. Introdução

Quem já teve algum contato com o GNU/Linux, mesmo que superficial, deve ter percebido a presença de vários diretórios (pastas) no sistema. Entretanto, eles estão organizados de uma forma talvez não muito familiar. Neste capítulo, vamos conhecer a organização e explorar a estrutura de diretórios de um sistema GNU/Linux.

Desde que o GNU/Linux foi criado, muito se tem feito para seguir um padrão em relação à estrutura de diretórios. O primeiro esforço para padronização de sistemas de arquivos para o GNU/Linux foi o ***FSSTND - Filesystem Standard***,

lançado no ano de 1994.

Cada diretório do sistema tem seus respectivos arquivos que são armazenados conforme regras definidas pela ***FHS - Filesystem Hierarchy Standard***, ou Hierarquia Padrão do Sistema de Arquivos, que define que tipo de arquivo deve ser guardado em cada diretório. Isso é muito importante, pois o padrão ajuda a manter compatibilidade entre as versões Linux existentes no mercado, permitindo que qualquer software escrito para o GNU/Linux seja executado em qualquer distribuição desenvolvida de acordo com os padrões FHS.

Atualmente, o FHS está na sua versão 2.3, e é mantido pelo *Free Standard Group*, uma organização sem fins lucrativos formada por grandes empresas como HP, IBM, Red Hat e Dell.

A FHS estar bem esclarecida, afinal é com ela que nós devemos fazer nossas atividades do dia-a-dia

3.3. Estrutura de Diretórios GNU/Linux

A estrutura de diretórios também é conhecida como ``Árvore de Diretórios'' porque tem a forma de uma árvore. Mas, antes de estudarmos a estrutura de diretórios, temos que ter em mente o que são diretórios.

Um diretório nada mais é do que o local onde os arquivos são guardados no sistema. O arquivo pode ser um texto, uma imagem, planilha, etc. Os arquivos devem ser identificados por nomes para que sejam localizados por quem deseja utilizá-los.

Um detalhe importante a ser observado é que o GNU/Linux é *case sensitive*, isto é, ele diferencia letras maiúsculas e minúsculas nos arquivos e diretórios.

Sendo assim, um arquivo chamado Arquivo é diferente de **ARQUIVO** e diferente de **arquivo**.

A árvore de diretórios do GNU/Linux tem a seguinte estrutura:

/								
bin	*cdrom*	*etc*	*lib*	*mnt*	*proc*	*root*	*var*	
boot	*dev*	*home*	*media*	*opt*	*sbin*	*srv*	*tmp*	*usr*

Da estrutura mostrada acima, o FHS determina que um sistema GNU/Linux deve conter obrigatoriamente 14 diretórios, especificados a seguir:

/ (raiz)

Este é o principal diretório do GNU/Linux, e é representado por uma ``/'' (barra). É no diretório raiz que ficam todos os demais diretórios do sistema.

Estes diretórios, que vamos conhecer agora, são chamados de subdiretórios pois estão dentro do diretório /.

```
/bin
```

O diretório /bin guarda os comandos essenciais para o funcionamento do sistema.

Esse é um diretório público, sendo assim, os comandos que estão nele podem ser utilizados por qualquer usuário do sistema. Entre os comandos, estão:

- bash;
- ls;
- echo;
- cp;

```
/boot
```

No diretório /boot estão os arquivos estáticos necessários à inicialização do sistema, e o gerenciador de boot.

O gerenciador de boot é um programa que carrega um sistema operacional e/ou permite escolher qual será iniciado.

```
/dev
```

No diretório /dev ficam todos os arquivos de dispositivos. O Linux faz a comunicação com os periféricos por meio de links especiais que ficam armazenados nesse diretório, facilitando assim o acesso aos mesmos.

```
/etc
```

No diretório /etc estão os arquivos de configuração do sistema. Nesse diretório vamos encontrar vários arquivos de configuração, tais como: scripts de inicialização do sistema, tabela do sistema de arquivos, configuração padrão para logins dos usuários, etc.

```
/lib
```

No diretório /lib estão as bibliotecas compartilhadas e módulos do kernel . As bibliotecas são funções que podem ser utilizadas por vários programas.

```
/media
```

Ponto de montagem para dispositivos removíveis, tais como:

- cd;
- dvd;
- disquete;
- pendrive;
- câmera digital;

Fique atento: Agora o diretório /media faz parte oficialmente das provas da LPI

```
/mnt
```

Esse diretório é utilizado para montagem temporária de sistemas de arquivos, tais como compartilhamentos de arquivos entre Windows e Linux, Linux e Linux, etc.

```
/opt
```

Normalmente, é utilizado por programas proprietários ou que não fazem parte oficialmente da distribuição.

```
/sbin
```

O diretório /sbin guarda os comandos utilizados para inicializar, reparar, restaurar e/ou recuperar o sistema. Isso quer dizer que esse diretório também é de comandos essenciais, mas os mesmos são utilizados apenas pelo usuário root.

Entre os comandos estão:

- halt
- ifconfig
- init
- iptables

```
/srv
```

Diretório para dados de serviços fornecidos pelo sistema cuja aplicação é de alcance geral, ou seja, os dados não são específicos de um usuário.

Por exemplo:

- /srv/www (servidor web)
- /srv/ftp (servidor ftp)

```
/tmp
```

Diretório para armazenamento de arquivos temporários. É utilizado principalmente para guardar pequenas informações que precisam estar em algum lugar até que a operação seja completada, como é o caso de um download.

Enquanto não for concluído, o arquivo fica registrado em /tmp, e, assim que é finalizado, é encaminhado para o local correto.

```
/usr
```

O diretório /usr contém programas que não são essenciais ao sistema e que seguem o padrão GNU/Linux, como, por exemplo, navegadores, gerenciadores de janelas, etc.

O diretório /usr é portável, perceba que dentro dele, existe praticamente uma outra arvore de diretórios independente da primeira, contendo, lib, bin, sbin dentre outras coisas.

```
/var
```

O diretório /var contém arquivos de dados variáveis. Por padrão, os programas que geram um arquivo de registro para consulta, mais conhecido como log, ficam armazenados nesse diretório. Além do log, os arquivos que estão aguardando em filas, também ficam localizados em /var/spool.

Os principais arquivos que se utilizam do diretório /var são :

- mensagens de e-mail;
- arquivos a serem impressos;

3.4. Diretório Recomendado

```
/proc
```

O /proc é um diretório virtual, mantido pelo kernel, onde encontramos a configuração atual do sistema, dados estatísticos, dispositivos já montados, interrupções, endereços e estados das portas físicas, dados sobre as redes, etc.

Aqui, temos subdiretórios com o nome que corresponde ao PID (Process ID) de cada processo.

Dentro deles, vamos encontrar diversos arquivos texto contendo várias informações sobre o respectivo processo em execução.

3.5. O diretório /sys

Pode-se dizer que esse diretório é um primo do diretório /proc. Dentro do diretório /sys podemos encontrar o quase o mesmo conteúdo do proc, mas de uma forma bem mais organizada para nós administradores.

Esse diretório está presente desde a versão 2.6 do kernel e traz novas funcionalidades o que se diz respeito a dispositivos PnP.

3.6. Diretórios Opcionais

Os diretórios /root e /home podem estar disponíveis no sistema, mas não precisam obrigatoriamente possuir este nome.

Por exemplo, o diretório /home poderia se chamar /casa, que não causaria nenhum impacto na estrutura do sistema.

```
/home
```

O /home contém os diretórios pessoais dos usuários cadastrados no sistema.

```
/root
```

Diretório pessoal do superusuário root.

O root é o administrador do sistema, e pode alterar a configuração (dele), configurar interfaces de rede, manipular usuários e grupos, alterar a prioridade dos processos, entre outras.

Dica: Utilize uma conta de usuário normal em vez da conta root para operar seu sistema.

Uma razão para evitar usar privilégios root é por causa da facilidade de se cometer danos irreparáveis como root; além do que, você pode ser enganado e rodar um programa Cavalo de Troia (programa que obtém poderes do super usuário) comprometendo a segurança do seu sistema sem que você saiba.

3.7. Comandos de Movimentação

Vamos aprender agora alguns comandos essenciais para a nossa movimentação dentro do sistema.

O comando pwd exibe o diretório corrente. Ele é muito útil quando estamos navegando pelo sistema e não lembramos qual o diretório atual.

```
# pwd
```

O comando cd é utilizado para mudar o diretório atual de onde o usuário está.

Ir para o diretório home do usuário logado:

```
# cd
# cd ~
```

Ir para o início da árvore de diretórios, ou seja, o diretório / :

```
# cd /
```

Ir para um diretório específico:

```
# cd /etc
```

Sobe um nível na árvore de diretórios:

```
# cd ..
```

Retorna ao diretório anterior:

```
# cd -
# ls
```

Entra em um diretório específico:

```
# cd /usr/include/X11
```

Sobe 2 níveis da árvore de diretórios

```
# cd ../../
```

Atenção! Note a diferença entre caminhos absolutos e relativos:
Absolutos: /etc/ppp; /usr/share/doc; /lib/modules
Relativos: etc/ppp; ../doc; ../../usr;

*Fique esperto para conhecer as diferenças entre o **.** e o **..** e o que eles representam para o sistema. Os comandos de movimentação muitas vezes são grandes alvos nas provas, uma boa interpretação desses comandos pode ser necessária, pois você pode precisar deles para resoler uma questão maior.*

3.8. Prática Dirigida

Através dos comandos: cd e pwd, navegue no sistema afim de explorar alguns diretórios.

1) Verificar o diretório atual:

```
$ pwd
```

2) Ir para o início da árvore de diretórios, ou seja, o diretório / :

```
$ cd /
```

3) Ir para o diretório home do usuário logado:

```
$ cd
$ cd ~
```

4) Ir para o diretório /usr/share:

```
$ cd /usr/share
```

5) Subir um nível na árvore de diretórios:

```
$ cd ..
```

6) Retornar ao diretório anterior:

```
$ cd -
```

7) Entre no diretório /var:

```
$ cd /var
```

8) Entre no diretório /etc e veja o resultado do comando pwd:

```
$ cd /etc
$ pwd
```

9) Utilize o comando cd .., para voltar um nível na hierarquia:

```
$ cd ..
```

10) Descubra em qual diretório você está através do comando pwd:

```
$ pwd
```

11) Utilize o comando cd ~, para voltar para seu diretório pessoal:

```
$ cd ~
```

12) Descubra em qual diretório você está através do comando pwd:

```
$ pwd
```

13) Utilize o comando cd -, para voltar ao ultimo diretório acessado:

```
$ cd -
```

14) Descubra em qual diretório você está através do comando pwd:

```
$ pwd
```

3.9. Exercício Teórico

1) Explore os diretórios abaixo, e escreva qual é a função de cada um deles. Justifique:

a) bin

____

____

b) boot

____

____

c) dev

____

____

d) etc

____

____

e) home

____

____

f) lib

____

____

g) media

____

____

h) mnt

____

____

i) var

____

____

j) opt

____

____

k) proc

____

____

l) root

____

m) sbin

n) srv

o) tmp

p) usr

2) Qual é a finalidade do comando pwd?

3) Escreva a função de cada um dos comandos abaixo:

a) cd -

b) cd ~

c) cd /

d) cd

e) cd ..

g) cd .

3.10. Laboratório

Alem de todos os diretórios listados acima, na raiz do sistema existe um diretório chamado lost+found, o que representa esse diretório?

Veja também:

FHS - *http://www.pathname.com/fhs/*

Free Standard Group - *http://www.linux-foundation.org/en/Main_Page*

Capítulo 4

Aprendendo comandos do GNU/Linux

4.1. Objetivos

- Criar e remover arquivos
- Criar e remover diretórios
- Criar Links

4.2. Introdução

Comandos são instruções passadas ao computador para executar uma determinada tarefa. No mundo *NIX (Linux,Unix), o conceito de comandos é diferente do padrão MS-DOS. Um comando é qualquer arquivo executável, que pode ser ou não criado pelo usuário.

Uma das tantas vantagens do Linux é a variedade de comandos que ele oferece, afinal, para quem conhece comandos, a administração do sistema acaba se tornando um processo mais rápido.

O shell é o responsável pela interação entre o usuário e o sistema operacional, interpretando os comandos.

É no shell que os comandos são executados.

4.2.1. Explorando o sistema

Veremos agora os comandos básicos para navegação no sistema.

O comando ls é utilizado para listar o conteúdo dos diretórios. Se não for especificado nenhum diretório, ele irá mostrar o conteúdo do diretório onde estamos no momento.

Lista o conteúdo do diretório atual:

```
# ls
```

4.3. O comando ls

O comando ls possui muitos parâmetros, veremos aqui as opções mais utilizadas. A primeira dela é o **-l** que lista os arquivos ou diretórios de uma forma bem detalhada (quem criou, data de criação, tamanho, dono e grupo a qual eles pertencem).

```
# ls -l /
drwxr-xr-x4 root root 1024 2007-01-15 23:17 boot
```

Veja que a saída desse comando é bem detalhada. Falando sobre os campos, para o primeiro caractere temos algumas opções:

d => indica que se trata de um diretório
l => indica que se trata de um link (como se fosse um atalho - também vamos falar sobre ele depois)
- => hífen, indica que se trata de um arquivo
c => indica dispositivo de caractere
b => indica dispositivo de bloco

O campo rwxr-xr-x lista as permissões, enquanto os campos root indicam quem é o usuário e grupo dono desse diretório que, no nosso caso, é o administrador do sistema, o root. O número antes do dono indica o número de hard links, um assunto abordado apenas em cursos mais avançados.

O campo 1024 indica o tamanho do arquivo, e o campo 2007-01-15 23:17 informa a data e hora em que o diretório foi criado. Finalmente, no último campo temos o nome do arquivo ou diretório listado, que, no nosso exemplo, é o boot.

Com relação a diretórios, é importante ressaltar que o tamanho mostrado não corresponde ao espaço ocupado pelo diretório e seus arquivos e subdiretórios. Esse espaço é aquele ocupado pela entrada no sistema de arquivos que corresponde ao diretório.

A opção a lista todos arquivos, inclusive os ocultos:

```
 # ls -a /root
..aptitude.bashrc.profile .rnd.ssh.vmware
.. .bash_history .kde .qt root_161206 .viminfo .Xauthority
```

Veja que, da saída do comando anterior, alguns arquivos são iniciados por . (ponto). Esses arquivos são ocultos.

No Linux, arquivos e diretórios ocultos são iniciados por um . (ponto).

Lista arquivos de forma recursiva, ou seja, lista também os subdiretórios que estão dentro do diretório /:

```
 # ls -R /
```

4.3.1. Coringas

O significado da palavra coringa no dicionário é o seguinte: carta de baralho, que em certos jogos, muda de valor e colocação na sequência. No sistema GNU/Linux é bem parecida a utilização desse recurso. Os coringas são utilizados para especificar um ou mais arquivos ou diretórios.

Eles podem substituir uma palavra completa ou somente uma letra, seja para listar, copiar, apagar, etc. São usados três tipos de coringas no GNU/Linux:

*** -** *Utilizado para um nome completo ou restante de um arquivo/diretório;*

? **-** *Esse coringa pode substituir uma ou mais letras em determinada posição;*

[padrão] **-** *É utilizado para referência a uma faixa de caracteres de um arquivo/diretório.*

[a-z][0-9] **-** *Usado para trabalhar com caracteres de a até z seguidos de um caractere de 0 até 9.*

[a,z][1,0] **-** *Usado para trabalhar com os caracteres a e z seguidos de um caractere 1 ou 0 naquela posição.*

[a-z,1,0] **-** *Faz referência do intervalo de caracteres de a até z ou 1 ou 0 naquela posição.*

A diferença do método de expansão dos demais, é que a existência do arquivo ou diretório é opcional para resultado final. Isto é útil para a criação de diretórios. Lembrando que os 3 tipos de coringas mais utilizados (``*,?,[ ]'') podem ser usados juntos. Vejamos alguns exemplos:

Supondo que existam 5 arquivos no diretório /home/usuário. Podemos listá-los:

```
# ls
arq1.txt arq2.txt arq3.txt arq4.new arq5.new
```

Vamos listar todos os arquivos do diretório /home/usuário. Podemos usar o coringa ``*'' para visualizar todos os arquivos do diretório:

```
# cd /home/usuário
# ls *
arq1.txt arq2.txt arq3.txt arq4.new arq5.new
```

Para listarmos todos os arquivos do diretório /home/usuário que tenham ``new'' no nome:

```
 # ls *new*
arq4.new arq5.new
```

No caso, o comando #ls /tmp/teste/* foi citado, mas não tem muito sentido utilizar esse comando, é importante ressaltar que a utilização do * se aplica para um

diretório cheio de arquivos, como mostrado no caso dois, utilizado para procurar o arquivo em específico.

4.3.2. Usando coringas no Shell

Listar todos os arquivos que começam com qualquer nome e terminam com .txt:

```
# ls *.txt
```

Listar todos os arquivos que começam com o nome arq, tenham qualquer caractere no lugar do coringa, e terminem com .txt:

```
# ls arq?.txt
```

Para listar todos os arquivos que começam com o nome arq, tenham qualquer caractere entre o número 1-3 no lugar da 4ª letra e terminem com .txt. Neste caso, se obtém uma filtragem mais exata, pois o coringa especifica qualquer caractere naquela posição e [ ] especifica números, letras ou intervalo que serão usados.

```
# ls arq[1-3].txt
```

Para listar somente arq4.new e arq5.new podemos usar os seguintes métodos:

```
# ls *.new
# ls *new*
# ls arq?.new
# ls arq[4,5].*
# ls arq[4,5].new
```

O parâmetro **-***i do ls, pode ter um grande valor quando o papo são os inodes.*

Existem muitas outras maneiras de fazer a mesma coisa mas depende muito de cada um que vai utilizar. A criatividade nesse momento conta muito. No exemplo

anterior, a última forma resulta na busca mais específica. O que pretendemos é mostrar como visualizar mais de um arquivo de uma só vez. O uso de coringas é útil para copiar arquivos, apagar, mover, renomear, e nas mais diversas partes do sistema.

4.4. Criação, movimentação, cópia e remoção de arquivos e diretórios

Para criar um arquivo, podemos simplesmente abrir um editor de texto e salvá-lo. Mas existem outras formas.

Uma das formas mais simples é usando o comando touch:

```
# touch arquivo
```

A grande maioria dos comandos básicos devem fazer parte da sua base sólida de conhecimento, é provável que você precise dele para resolver problemas maiores.

O comando **mkdir** é utilizado para criar um diretório no sistema. Um diretório é como uma pasta onde você guarda seus arquivos.

Exemplo:

Cria o diretório yago:

```
# mkdir yago
```

Cria o diretório 4linux e o subdiretório alunos:

```
# mkdir -p 4linux/alunos
```

A opção **-p** irá criar o diretório 4linux e o subdiretório alunos, caso não existam.

O comando **rm** é utilizado para apagar arquivos, diretórios e subdiretórios que estejam vazios ou que contenham arquivos.

Exemplos:

Remove o arquivo teste.txt:

```
# rm teste.txt
```

Remove o arquivo yago.txt pedindo confirmação:

```
# rm -i yago.txt
rm: remove arquivo comum `yago.txt'? y
```

A opção **-i** solicita a confirmação para remover o arquivo yago.txt.

Remove o diretório 4linux:

```
# rm -r 4linux
```

A opção **-r** é recursivo, ou seja, irá remover o diretório 4linux e o seu conteúdo.

Observação: Muita atenção ao usar o comando rm! Uma vez que os arquivos e diretórios são removidos não podem mais ser recuperados!

O comando **rmdir** é utilizado para remover diretórios vazios.

Exemplos:

Remove o diretório yago:

```
# rmdir yago
```

Remove o diretório 4linux e o subdiretório alunos:

```
# rmdir -p hackerteen/alunos
```

O comando mv serve tanto para renomear um arquivo quanto para movê-lo:

```
# mv arquivo caminho/diretório-destino/
# mv arquivo novo-nome
# mv diretório novo-nome
# mv diretório caminho/diretório-destino/
```

O comando **cp** serve para fazer cópias de arquivos e diretórios:

```
# cp arquivo-origem arquivo-destino
# cp arquivo-origem caminho/diretório-destino/
# cp -R diretório-origem nome-destino
# cp -R diretório-origem caminho/diretório-destino/
```

Um opção do comando cp muito útil no nosso dia-a-dia é o -p, o que faz com que o cp mantenha o timestamp dos arquivos, assim não modificando seus donos nem suas permissões.

4.5. Prática Dirigida

1) Listar o conteúdo do diretório /:

```
# ls /
```

2) Listar o conteúdo do diretório /root em formato longo:

```
# ls -l /root/
```

3) Listar somente o diretório /boot em formato longo:

```
# ls -ld /boot/
```

4) Listar todos os arquivos do diretório /root, inclusive os ocultos:

```
# ls -a /root
```

5) Listar o conteúdo do diretório /boot de forma recursiva:

```
# ls -R /boot/
```

6) Criar o diretório estudo dentro do diretório /tmp:

```
# mkdir /tmp/estudo
```

7) Criar a seguinte estrutura de diretórios: /backup/2007/fevereiro

```
# mkdir -p /backup/2007/fevereiro
```

8) Remover o diretório /tmp/estudo utilizando o comando rmdir:

```
# rmdir /tmp/estudo
```

9) Crie os arquivos estudo.txt e alunos.txt dentro de /backup/2007/fevereiro.

```
# touch /backup/2007/fevereiro/estudo.txt
# touch /backup/2007/fevereiro/alunos.txt
```

10) Entre no diretório /backup/2007/fevereiro e copie o arquivo estudo.txt para aula.txt:

```
# cd /backup/2007/fevereiro
# cp estudo.txt aula.txt
```

11) Copie o diretório /backup/2007/fevereiro para /backup/2007/janeiro:

```
# cp -R /backup/2007/fevereiro /backup/2007/janeiro
```

12) Remova o arquivo estudo.txt do diretório /backup/2007/fevereiro: # cd /backup/2007/fevereiro

```
# rm estudo.txt
```

13) Renomeie o arquivo alunos.txt do diretório /backup/2007/fevereiro:

```
# cd /backup/2007/fevereiro
# mv alunos.txt teste.txt
```

14) Mova o diretório /backup/2007/fevereiro para /backup/2007/abril:

```
 # mv /backup/2007/fevereiro /backup/2007/abril
```

15) Utilize o comando stat para descobrir algumas informações importantes:

```
#stat /backup
```

4.6. Exercício Teórico

1) Explique com suas palavras o que é um inode?

____

____

____

____

2) Qual é o comando completo a ser executado para criarmos a estrutura de diretórios /stone/blue/gold?

____

____

____

____

3) Qual a função do comando rmdir?

____

____

____

____

4) Por que não devemos executar o comando rm com as flags -r e -f?

____

____

____

____

5) Qual é a opção do comando cp que copiaria arquivos de um diretório de forma recursiva?

____

____

____

____

6) Qual a função do comando mv?

____

____

____

____

7) Qual a função do comando ln?

____

____

____

____

8) Você é um estagiário muito organizado. Ao checar os arquivos do servidor, percebe que alguns arquivos de configuração e arquivos de log estão espalhados no diretório /root. Quais deveriam ser os diretórios corretos para armazenar esse tipo de arquivo?

____

____

____

____

4.7. Laboratório

1) Liste os arquivos que terminam com a palavra .conf dentro do diretório /etc;
2) Busque no diretório raiz [/] todos os diretórios que terminem com a letra ``n'';

Capítulo 5

Comandos úteis de linha de comando

5.1. Objetivos

- Conhecer alguns comandos importantes para o dia-a-dia.

No mundo GNU/Linux, a maioria das operações são realizadas por meio de comandos escritos. Em geral, permitem um maior controle e flexibilidade de operações, além de poderem ser incluídos em scripts. Neste capítulo iremos aprender sobre alguns comandos básicos.

5.2. Trabalhando com entrada e saída de dados

Esta parte é extremamente importante, pois se trabalha bastante com isso. Por padrão, a entrada do Shell é o teclado, a saída, a tela, e os erros são exibidos na tela também.

Os termos geralmente usados são:

- Entrada de dados, representada por stdin;
- Saída de dados, representada por stdout;
- Saída de erros, representada por stderr;

Mas isso pode ser mudado com o uso de caracteres de redirecionamento, veja abaixo:

Mas isso pode ser mudado com o uso de caracteres de redirecionamento, veja abaixo:

Para mudar saída padrão:

- > - Redireciona a saída em um arquivo apagando o conteúdo anterior (se existir); Exemplo:

```
# ls / > tst
# cat tst
# ls /var > tst
# cat tst
```

- >> - Redireciona a saída no final de um arquivo, preservando-o;

Exemplo:

```
 # ls / >> tst
# cat tst
# ls /var >> tst
# cat tst
```

Comandos auxiliares:

- **|** (pipe, pronuncia-se paipe): Serve para canalizar saída de dado para outro comando;

5.3. Comandos para paginação

Quando os arquivos são maiores do que a altura da tela do computador, eles são mostrados sequencialmente e até seu final. Isso ocorre numa velocidade que impede que se consiga ler algo. Para esse tipo de arquivo grande, usamos os comandos de paginação. Eles controlam a maneira que os dados de um arquivo são exibidos, seja permitindo uma navegação elementar ou permitindo atingir porções específicas de um arquivo.

5.3.1. Mostrando o conteúdo e/ou concatenando

O comando cat pode ser utilizado para mostrar o conteúdo de um arquivo. Por exemplo, o comando abaixo mostra o conteúdo do arquivo teste.dat.

```
# cat teste.dat
```

Mas o comando cat pode ser utilizado também para concatenação de arquivos. No primeiro exemplo abaixo, é mostrado na tela o conteúdo dos arquivos teste.dat aux.dat. No segundo exemplo, usamos o operador de redirecionamento da saída padrão de modo que a saída do comando cat seja gravada no arquivo tudo.dat.

```
 # cat teste.dat aux.dat
# cat teste.dat aux.dat > tudo.dat
```

De forma análoga, o comando tac também serve para mostrar o conteúdo e concatenar arquivos. Porém, ele mostra o conteúdo de forma reversa, linha a linha. Em outras palavras, ele imprime primeiramente a última linha do arquivo especificado, e finaliza imprimindo a primeira. O exemplo abaixo, mostra o uso do tac

para mostrar o conteúdo reverso do arquivo teste.dat.

```
# tac teste.dat
```

5.3.2. Controlar o fluxo: more e less

Para que a leitura de um arquivo grande na linha de comando seja possível, podemos usar um editor de textos como o vi ou emacs. Contudo, para uma leitura rápida na linha de comando podemos usar os comandos more e less. O comando more permite a leitura contínua de um arquivo. Sempre que a tela é preenchida, o comando more espera por uma ação do usuário para mostrar mais conteúdo.

Pressionando ENTER uma linha a mais é mostrada, pressionando a barra de espaços uma nova página é mostrada. Não é possível retornar (subir) usando o comando more.

```
# more /var/log/syslog
```

O comando less é mais sofisticado e permite ir e voltar na leitura de um arquivo.

```
# less /var/log/syslog
```

5.3.3. Porções específicas: head e tail

Freqüentemente, queremos ter acesso a porções específicas de um arquivo. Às vezes queremos apenas as linhas iniciais ou as linhas finais. E às vezes queremos um pedaço definido do arquivo. Para essas necessidades, usamos os comandos head para ler porções superiores de um arquivo e tail para ler as porções inferiores. Para ler as 10 primeiras linhas de um arquivo, podemos usar:

```
# head /var/log/syslog
```

Para ler as 10 últimas linhas de um arquivo, podemos usar:

```
# tail /var/log/syslog
```

Os comandos podem ser combinados usando o | (lê-se pipe). Por exemplo, para ler o pedaço entre as linhas 20 e 40 de um arquivo, podemos usar:

```
# head -n 40 /var/log/syslog | tail -n 20
```

O comando acima lê as 40 primeiras linhas do arquivo /var/log/syslog que são passadas para o comando tail que retorna as 20 últimas linhas deste intervalo (as 20 últimas das 40 primeiras = da 20 a 40).

*Apesar de parecerem simples, os comandos head e tail fornecem opções de valiosa utilidade. Um grande exemplo é usar a opção **-f** no tail para verificar logs em tempo real.*

5.3.4. Contagem: wc

Grande parte dos arquivos de configuração e de dados usa uma linha por registro. A contagem destas linhas pode nos fornecer informações muito interessantes.

Por exemplo, a saída abaixo:

```
# wc /etc/passwd
```

Indica que o arquivo contém 32 linhas, 49 blocos (palavras) e 1528 caracteres.

Caso seja necessário apenas o número de linhas, o comando wc pode ser usado com o parâmetro -l, como abaixo:

```
# wc -l /etc/passwd
```

Outros parâmetros possíveis são -w para blocos (palavras) e -c para caracteres.

5.3.5. Classificação: sort

Para diversas ações como eliminação de itens repetidos e rápida visualização de nomes é interessante que possamos classificar um arquivo texto. Na linha de comando, os arquivos textos podem ser classificados usando o comando sort.

A saída do comando abaixo não segue a ordem alfabética:

```
# cat /etc/passwd
```

Podemos mostrar a saída classificada em ordem alfabética, como abaixo:

```
# sort /etc/passwd
```

O comando sort pode ser modificado usando os parâmetros:

- **-f** não considera se as letras estão em caixa alta ou baixa;
- **-n** classificação numérica;
- **-r** classifica na ordem invertida.

5.3.6. Mostrar algo: echo

O comando echo é usado para ecoar algo na tela ou direcionado para um arquivo. Isso é bastante útil para automação. Na linha de comando o echo é útil para inspecionar variáveis de ambiente, que são parâmetros guardados em memória e que definem o ambiente em uso.

Por exemplo, para saber qual a pasta pessoal definida em $HOME do usuário atual:

```
# echo $HOME
```

Para saber qual o idioma definido no console:

```
# echo $LANG
```

Usando o caractere de redirecionamento >, podemos enviar a saída do comando echo para outro destino:

```
# echo $LANG > /tmp/teste
# cat /tmp/teste
```

No exemplo acima, o arquivo teste contém o valor da variável de ambiente $LANG.

5.4. Filtragem

Uma necessidade constante dos administradores é encontrar informações dentro dos arquivos. Para ilustrar, podemos localizar o texto bash no arquivo /etc/passwd:

```
# grep bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
saito:x:1000:1000:saito,,,:/home/saito:/bin/bash
postgres:x:108:113:PostgreSQL
administrator,,,:/var/lib/postgresql:/bin/bash
jboss:x:1001:1001:JBoss Administrator,,,:/home/jboss:/bin/bash
```

Outra situação possível é procurar pelas entradas que não possuem bash no arquivo passwd. Para isso, usamos o parâmetro -v (inVerter), que inverte a filtragem do grep:

```
# grep -v bash /etc/passwd
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
```

Outros parâmetros do comando grep:

- **-A** [n] Mostra n linhas depois;
- **-B** [n] Mostra n linhas antes;
- **-h** Omite o nome do arquivo nas buscas;
- **-i** Ignora diferença entre maiúsculas e minúsculas;
- **-n** Mostra o número de cada linha encontrada;
- **-v** Inverte a busca, ou seja, encontra apenas as linhas onde o padrão não existir.

O grep pode ser combinado com a saída de outros comandos com o uso do | (pipe). Por exemplo, no comando abaixo, o grep filtra as linhas de endereços IP da saída do comando ifconfig.

```
# ifconfig | grep end.:
```

E, a seguir, o grep é aplicado para filtrar os últimos usuários logados no primeiro terminal (tty1):

```
# last |grep tty1
root tty1 Thu Feb 22 12:19 - 14:21 (02:01)
root tty1 Thu Feb 22 10:50 - down(00:00)
```

Alguns outros tipos de filtros como o grep também podem ser encontrados - ngrep e pgrep.

5.4.1. Filtrar colunas: cut

O comando cut pode ser muito útil para conseguir listagens a partir de arquivos com separadores de colunas definidos.

Por exemplo, para conseguir a primeira coluna do arquivo /etc/passwd, cujo delimitador de colunas é o sinal :, podemos usar o comando:

```
 # cut -f1 -d: /etc/passwd
root
daemon
bin
```

O comando awk é um primo do cut, mas possui mais recursos e opções para expressões regulares.

5.4.2. Determinando o tipo de arquivo: file

No Linux, extensões de arquivos têm apenas a função de nos auxiliar a nomear os arquivos, a identificá-los e organizá-los facilmente. Não é a extensão que determina o tipo do arquivo, mas sim o seu conteúdo. Por exemplo, se renomearmos um arquivo imagem de 4Linux.jpg para 4Linux.html, ele continuará sendo um arquivo de imagem JPEG.

O comando file determina o tipo do arquivo analisando o seu próprio conteúdo. O exemplo abaixo mostra o uso deste comando:

```
# file arquivo
```

5.5. Administrativos

5.5.1. Espaço em Disco

Aproxima para a unidade de medida mais próxima, mais legível para o ser humano.

```
 $ df -h <arquivo/diretório/partição>
```

Mostra em kilobytes.

```
 $ df -k <arquivo, diretório ou partição>
```

Mostra em Megabytes.

```
$ df -m <arquivo, diretório ou partição>
```

5.5.2. Definindo tamanho dos objetos

```
$ du -h <arquivo, diretório ou partição>
```

Aproxima para a unidade de medida mais próxima, mais legível para o ser humano.

```
$ du -b <arquivo, diretório ou partição>
```

Mostra em bytes.

```
$ du -k <arquivo, diretório ou partição>
```

Mostra em kilobytes.

```
$ du -m <arquivo, diretório ou partição>
```

Mostra em Megabytes.

```
$ du -l <arquivo, diretório ou partição>
```

Mostra a quantidade de links que arquivo/diretório/partição tem.

```
$ du -s <arquivo, diretório ou partição>
```

Modo silencioso, ou seja, não mostra subdiretórios.

5.5.3. Mostrar o uso de memória RAM: free

O comando free mostra o consumo de memória RAM e os detalhes sobre uso de memória virtual (SWAP):

```
# free
```

A saída do comando será:

```
 total used free shared buffers cached
Mem: 2066856 950944 1115912 038920 342612
-/+ buffers/cache: 569412 1497444
Swap: 570268 0 570268
```

5.5.4. Mostrar e/ou ajustar a data do sistema: date

O comando date pode ser utilizado para mostrar a data e a hora do sistema, e também para ajustá-las. Há várias formas de se utilizar esse comando. A primeira delas é a mais simples:

```
 # date
```

Esse comando mostra a data e a hora atuais do sistema numa formatação padrão. Pode-se utilizar uma string como parâmetro para formatar a saída. O exemplo abaixo mostra o uso de uma string de formatação e o resultado a seguir. Mais informações sobre as opções de formatação podem ser encontradas nas páginas do manual.

```
 # date +"%H:%M, %d-%m-%Y"
12:44, 27-06-2007
```

A outra utilidade do comando date é ajustar a hora do sistema. Obviamente, isso pode ser feito apenas

pelo usuário administrador. A sintaxe é:

```
 # date mmddHHMMYYYY
```

Onde:

- mm - número do mês;
- dd - dia do mês;

- HH – hora;
- MM – minutos;
- YYYY – ano.;

5.5.5. Mostrar por quanto tempo o computador está ligado: uptime

O comando uptime mostra por quanto tempo o computador está ligado. Além disso, mostra informações sobre o uso do processador:

```
# uptime
```

A saída do comando será:

```
03:20:37 up 16:35, 3 users, load average: 0.16, 0.27, 0.33
```

5.5.6. Mostrar informações sobre o sistema: uname

O comando uname pode ser usado para mostrar informações sobre a versão do kernel em uso e a arquitetura:

```
# uname -a
```

A saída do comando será:

```
Linux professor 2.6.18-3-686 #1 SMP Mon Dec 4 16:41:14 UTC 2006 i686
GNU/Linux
```

5.5.7. Diferença entre arquivos: diff

O programa diff nos permite verificar a diferença entre arquivos e diretórios. No caso de diretórios, é importante o uso da opção -r para assegurar a comparação de todos os subdiretórios.

```
# diff arquivo1 arquivo2
# diff -r dir1 dir2
```

5.5.8. Tempo de execução de um programa: time

O comando time permite medir o tempo de execução de um programa. Sua sintaxe é:

```
# time programa
```

5.5.9. Localização no sistema: find

O comando find procura por arquivos/diretórios no disco. Ele pode procurar arquivos pela sua data de modificação, tamanho, etc, com o uso de opções. Find, ao contrário de outros programas, usa opções longas por meio de um ``-''.

Sintaxe:

find [diretório] [opções/expressão]

- **-name [expressão] :**

Procura pelo nome [expressão] nos nomes de arquivos e diretórios processados.

```
# find /etc -name *.conf
```

- **-maxdepth [num] :**

Limite a profundidade de busca na árvore de diretórios. Por exemplo, limitando a 1, irá procurar apenas no diretório especificado e não irá incluir nenhum subdiretório.

```
# find /etc -maxdepth 1 -name *.conf
```

- **-amin [num] :**

Procura por arquivos que foram acessados [num] minutos atrás. Caso seja antecedido por ``-'', procura por arquivos que foram acessados entre [num] minutos atrás e o momento atual.

```
# find ~ -amin -5
```

- **-atime [num] :**

Procura por arquivos que foram acessados [num] dias atrás. Caso seja antecedido por ``-``, procura por arquivos que foram acessados entre [num] dias atrás e a data atual.

```
# find ~ -atime -10
```

- **-uid [num] :**

Procura por arquivos que possuem a identificação numérica do usuário igual a [num].

```
# find / -uid 1000
```

- **-user [nome] :**

Procura por arquivos que possuem a identificação de nome do usuário igual a [nome].

```
# find / -user aluno
```

- **-perm [modo] :**

Procura por arquivos que possuem os modos de permissão [modo]. Os [modo] de permissão podem ser numérico (octal) ou literal.

```
# find / -perm 644
```

- **-size [num] :**

Procura por arquivos que tenham o tamanho [num]. O tamanho é especificado em bytes. Você pode usar os sufixos k, M ou G para representar em quilobytes, Megabytes ou Gigabytes. [num] Pode ser antecedido de ``+'' ou ``-'' para especificar um arquivo maior ou menor que [num].

```
# find / -size +1M
```

- **-type [tipo] :**

Procura por arquivos do [tipo] especificado. Os seguintes tipos são aceitos:

- **b -** bloco
- **c** - caractere
- **d -** diretório
- **p** - pipe
- **f -** arquivo regular
- **l-** link simbólico
- **s -** socket

```
# find /dev -type b
```

Outros exemplos:

```
# find / -name grep
```

Procura no diretório raiz e nos subdiretórios um arquivo/diretório chamado grep.

```
# find / -name grep -maxdepth 3
```

Procura no diretório raiz e nos subdiretórios até o 3º nível, um arquivo/diretório chamado grep.

```
# find . -size +1000k
```

Procura no diretório atual e nos subdiretórios um arquivo com tamanho maior que 1000 kbytes (1Mbyte).

```
# find / -mmin -10
```

Procura no diretório raiz e nos subdiretórios um arquivo que foi modificado há 10 minutos atrás ou menos.

5.5.10. Localização usando base de dados: locate

O comando locate é um comando rápido de busca de arquivos, porém não usa busca recursiva na sua árvore de diretórios. Ele cria uma base de dados para que a busca seja mais rápida pelo comando updatedb, que inclusive poderá ser agendado para que a base de dados esteja sempre atualizada.

Para utilizá-lo, primeiro é necessário criar a sua base de dados usando a seguinte sintaxe:

```
 # updatedb
```

Quando esse comando é executado pela primeira vez costuma demorar um pouco. Para o comando locate, usamos a seguinte sintaxe:

```
 # locate howto
```

A saída do comando será:

```
/root/Documentação/securing-debian-howto.en.pdf
/usr/share/doc/diveintopython/html/appendix/fdl_howto.html
/usr/share/doc/cacti/html/graph_howto.html
```

```
/usr/share/doc/pcmciautils/mini-howto.txt.gz
/usr/share/doc/python2.4-xml/examples/test/output/test_howto
/usr/share/doc/python2.4-xml/examples/test/test_howto.py.gz
/usr/share/doc/python2.4-xml/howto.cls
/usr/share/doc/python2.4-xml/xml-howto.tex.gz
/usr/share/doc/python2.4-xml/xml-howto.txt.gz
/usr/share/vim/vim64/doc/howto.txt
```

5.6. Mais e mais comandos

- **awk** - linguagem de procura de padrões e processamento;
- **logger** comando de interface entre a shell e o syslog;
- **sed** editor de texto linha a linha;
- **seq** - imprime uma sequencia de números;
- **sleep** - insere uma pausa pelo número de segundos especificado;
- **expand** - converte tabs em espaços;
- **unexpand** - converte espaços em tabs;
- **join -** junta a saída de uma arquivo em outro arquivo, jogando na tela;
- **nl** - numera as linhas na saída do comando;
- **paste** - junta os arquivos na saída padrão;
- **split** - usado para dividir determinado arquivo em pedaços menores;
- **tr** - substitui ou remove os caracteres selecionados da entrada padrão para a saída padrão;
- **uniq** - remove linhas desnecessárias ou duplicadas, ou seja, ele faz uma espécie de listagem de cada linha única do arquivo;
- **fmt** - formatador de texto, muito prático quando precisa-se fazer uma formatação rápida em algum arquivo;
- **hexdump** - converte arquivo para hexadecimal, decimal, ASCII;
- **od** - converte arquivo para octal;
- **xargs** - poderoso, e muito bom para listagem de arquivos.

Dica LPI: Atenção:

A maioria desses comandos são bastante simples de ser utilizados e possuem poucos parâmetros, entretanto alguns como awk, sed, find, grep, xargs, uniq, join, paste e cut possuem manuais bastante extensos dada a quantidade de tarefas que podem realizar. Dessa forma a leitura de suas páginas de man são altamente sugeridas.

Os comandos e estrutura de shell script aqui citados são utilizados também no Red Hat.

5.7. Prática Dirigida

Inicialmente vamos executar exemplos de funcionamento dos comandos mais utilizados em Shell Scripts.

1) Copie o arquivo passwd do sistema para o diretório /tmp:

```
# cp /etc/passwd /tmp
```

2) Entre no diretório /tmp para iniciarmos os testes:

```
# cd /tmp
```

3) Agora, dentro do diretório /tmp, utilizaremos o awk para separar o conteúdo do arquivo passwd utilizando o caracter ``:'' como separador e imprimir na tela o usuário e sua respectiva shell adicionando a palavra ``uses'' entre uma coluna e outra:

```
# awk -F : '{print $1 " uses " $7}' passwd
```

4) Utilizemos o comando cat para ver o conteúdo do arquivo passwd:

```
# cat passwd
```

5) Com o comando cut vamos realizar um procedimento análogo ao do awk; entretanto com este comando não é possível inserir a palavra ``uses'' entre as colunas:

```
# cut -d : -f 1,7 passwd
```

6) Utilizemos o comando date para listar a hora atual do sistema. Em seguida vamos alterá-la e depois modificar o seu padrão de saída:

```
# date
# date mmddHHMMYYYY
# date +%Y%m%d-%H%M
```

7) Determine quanto cada filesystem está utilizando de seu espaço. Depois imprima em formato ``humam readable'', ou seja, que um humano entende fácilmente, utilizando o parâmetro -h. Na sequencia verifique qual a porcentagem de inodes utilizados:

```
# df
# df -h
# df -i
```

8) A única forma de determinar quanto espaço em disco um diretório está ocupando é somando o tamanho de todos os arquivos e arquivos em seus subdiretórios. O comando du realiza essa tarefa:

```
# du -h
```

9) Para imprimir uma mensagem na tela, basta utilizar o comado echo. Os parâmetros -ne fazem com que o echo interprete caracteres de controle como ;

```
# echo -ne "Um tab\tseuguido de quebra de linha\n"
```

10) Juntando dois arquivos em um:

```
 # echo "1:Debian Sarge:3.1:Stable" > sarge
# echo "1:Debian Etch:4.0:Stable" > etch
# join -t: sarge etch
```

O comando join (unir) concatena registros de dois arquivos de texto baseado em índices comuns entre os registros. No nosso caso o separador de campos será o caractere dois-pontos (-t:).

11) Comando paste, junta os arquivos na saída padrão. Diferente do join, ele joga os dois arquivos lado-a-lado.:

```
 # paste sarge etch
```

Ainda com o paste podemos, usar o parâmetro -d, de delimitador:

```
 # paste -d@ sarge etch
```

E também, um em baixo do outro:

```
 # paste -s sarge etch
```

12) Vamos procurar por arquivos que satisfaçam o padrão pas* dentro do diretório /etc:

```
 # find /etc -name 'pas*'
```

13) Realize a procura dentro do arquivo passwd por todas as linhas que iniciam pelo caracter ``s'':

```
 # grep ^s passwd
```

14) Mostre na tela apenas as primeira 15 linhas do arquivo passwd:

```
# head -n 15 passwd
```

15) O comando split é usado para dividir um arquivo em pedaços menores, muito útil quando se tem dois disquetes e um arquivo de 2 Mb:

```
# cp /var/log/messages .
# split --lines=50 messages
```

Isso irá gerar X arquivos com 50 linhas cada:

```
# wc -l x*
```

Apague todos os arquivos:

```
# rm -r x*
```

16) Outro teste para nós fazermos:

```
# tar cvzf backup.sbin.tar.gz /sbin/
# du -sh /sbin/
# du -sh backup.sbin.tar.gz
```

Feito o backup e conferido os tamanhos, vamos agora utilizar o comando split:

```
# split --bytes=1048576 backup.sbin.tar.gz
```

Que irá gerar X arquivos com 1 Mb cada.

Onde:

```
1048576=1024*1024*1
```

Ou seja, 1Mb corresponde a 1048576 bytes.

Para conferir:

```
# du -sh x*
```

17) Vamos verificar que a linha foi escrita mostrando as últimas 15 linhas do respectivo arquivo de log:

```
# tail -n 15 /var/log/messages
# tail -n 15 /var/log/syslog
```

Mostre na tela como ficaria o arquivo passwd se substituíssemos todos os

caracteres ``:'' pelo padrão ``_MU_'':

```
# sed 's/:/_MU_/g' passwd
```

18) Utilize o comando seq para imprimir a sequencia de números de 1 a 100 pulando de dois em dois:

```
# seq 1 2 100
```

19) Faça a shell aguardar por dez segundos...

```
# sleep 10
```

20) Ordene as linhas do arquivo passwd:

```
# sort passwd
```

21) Vamos substituir e remover os caracteres do arquivo:

```
# tr a-z A-Z < /etc/passwd
# tr -d 0 < /etc/passwd
```

22) Quantas linhas há no arquivo passwd?

```
 # wc -l passwd
```

23) Verifique os usuários que estão logados no sistema:

```
 # who
```

24) Vamos agora listar diretórios utilizando o xargs:

```
# ls / | xargs -n1
# ls / | xargs -n2
# ls / | xargs -n3
```

Você percebeu que no primeiro comando ele listou o diretório, jogando na tela um nome de cada vez. O segundo comando fara o mesmo só que com dois nomes na mesma linha, e o terceiro também.

Outros testes com o xargs:

```
# ls / > teste_xargs.txt
# cat teste_xargs.txt
# cat teste_xargs.txt | xargs -n 2
# xargs -n 3 < teste_xargs.txt
```

25) Comando nl, numera as linhas na saída do comando:

```
# nl /etc/passwd
# grep sys /etc/passwd | nl
# ls -l /etc | nl
```

```
# ls -l /etc | tail | nl
```

5.8. Exercícios Teóricos

1) Qual é a principal diferença entre os comandos find e locate?

____
____
____
____

2) Qual é a função do comando tac?

____
____
____
____

3) Qual é a função do comando echo?

____
____
____
____

4) Qual é a diferença entre o comando du e o comando df?

____
____
____
____

5) Qual é a utilidade do comando time?

____
____
____
____

5.9. Laboratório

1) Obtenha uma lista de usuários a partir do arquivo /etc/passwd, contendo as colunas referentes ao usuário e seu diretório pessoal;
2) Obtenha uma lista classificada por tamanho dos arquivos da pasta /var/log.

Capítulo 6

Conhecendo a Documentação

6.1. Objetivos

- Diferenciar how-to, manual e documentação;
- Localizar os meios de ajuda internos do sistema;
- Descobrir aonde podemos buscar documentação na internet;

6.2. Introdução Teórica

Hoje em dia, não basta conhecermos bem alguma coisa. Especialmente nas áreas de tecnologia, devido a sua constante evolução, é preciso saber aprender para manter-nos atualizados. Neste capítulo, vamos aprender a consultar as documentações existentes e como buscar informações sobre o que precisamos.

O Sistema Operacional GNU/Linux possui uma vasta biblioteca de documentação. Antes de recorrermos a ajuda de outras pessoas, devemos lembrar que podemos ter a respostas que precisamos no próprio sistema, bem a nossa frente,

ao teclar de um simples comando. Essa documentação em grande parte dos casos é de extrema qualidade.

O GNU/Linux cresceu porque a comunidade que contribui para o sistema e sua documentação não tem medo ou receio de compartilhar informações e coloca o que foi desenvolvido no próprio sistema. É muito importante reforçar que no software livre, as pessoas nunca ocultam seu know-how, ou seja, você pode perguntar a vontade, desde que saiba perguntar e aonde perguntar.

A documentação do GNU/Linux pode ser vista também como fonte de conhecimento, aonde pessoas podem aprender muito sobre cada um dos serviços tanto na parte prática quanto na parte teórica.

Essa ajuda é provida por meio dos manuais, as famosas Man Pages, outros comandos que podem nos informar de maneira rápida os parâmetros que podemos utilizar e pelos How-Tos que cada aplicação instalada pode nos fornecer.

Toda essa documentação que possuímos no sistema GNU/Linux está disponível no site: ***http://www.tldp.org (The Linux Documentation Project),*** *o site oficial de documentações sobre GNU/Linux.*
Um diferencial deste site, é ter a documentação em vários idiomas e formatos (pdf, html, txt e outros).

Abaixo vamos começar a entender o que significa documentação e as formas que a documentação é apresentada;

6.3. Formas de Documentação

Existem diversas formas de se documentar um projeto, dentre elas temos How-to's, manuais e documentações.

6.3.1. How-to's

Os How-to's são documentos que focam uma necessidade específica, como montar um firewall, instalar uma webcam, configurar placas de som, configurar um

servidor web e muitos outros. Normalmente esses documentos são instalados junto com suas respectivas aplicações, ou em algumas vezes existe um pacote específico para a documentação daquela aplicação. Os how-to's também são conhecidos como cook-books.

O diretório de How-to's do GNU/Linux é o /usr/share/ doc. Por exemplo, se desejamos saber como configurar um firewall, podemos consultar os arquivos do diretório:

```
# cd /usr/share/doc/iptables/html/
```

Na internet existem diversos sites de how-to's para linux, dentre eles o mais conhecido no brasil é o VivaoLinux, conhecido também como VOL:

- *http://www.vivaolinux.com.br*

Muitas vezes o uso de how-to's ou cook-book's, não agrega um bom conhecimento, e somente uma lista de afazeres para chegar a um objetivo. Quando o software é atualizado, todo aquele conhecimento fica dependente de um outro how-to's.

6.3.2. Manuais

Diferente dos How-tos os manuais não vão te mostrar um passo a passo ou mesmo te dar uma lista de afazeres, o principal objetivo do manual é te mostrar como as funcionalidades daquele software podem ser usadas. Com o manual o aprendizado para a utilização da ferramenta é facilitado, já que o mesmo possui alguns exemplos de usabilidade.

Esses manuais podem ser encontrados através do comando man, o qual veremos ainda nesse capítulo, um pouco mais a frente.

6.3.3. Documentação

A palavra documentação é muito intensa, quando falamos em documentar uma ferramenta ou qualquer outra coisa, estamos na realidade abrangendo uma série de outros itens importantes, dentre eles os How-to's e os manuais. Com a

documentação de um projeto é possível entender absolutamente TUDO sobre o mesmo, ou seja, essa documentação deve mostrar todas as partes relacionadas ao projeto.

Podemos por exemplo citar a documentação de um projeto de rede, aonde nela deve constar não só documentos como how-to's e manuais, mas sim todas as especificações dos componentes, bem como cabos, switch's e routers entre milhares de outros detalhes muito importantes.

Como esse tipo de documentação é muito especifica, devemos consultar o site de cada projeto individualmente caso desejarmos obter essa fonte de conhecimento.

6.4. Comandos de ajuda

Existem diversos comandos de ajuda no GNU/Linux, vamos abordar cada um deles logo abaixo:

6.4.1. Comando help

O comando help provê ajuda para comandos internos do interpretador de comandos, ou seja o comando help fornece ajuda rápida; é útil para saber que opções podem ser usadas com os comandos internos do interpretador de comandos (shell).

Para visualizar uma ajuda rápida para todos os comandos internos do sistema, podemos fazer da seguinte forma:

```
# help
```

Com esse comando também podemos descobrir quais são os comandos internos do interpretados de comandos.

Caso desejemos visualizar a ajuda rápida para somente um comando interno, usamos esta outra sintaxe:

```
# help [comando]
```

O comando help somente mostra a ajuda para comandos internos.

Para comandos externos, o help aparece como parâmetro. Por exemplo:

```
sleep --help
```

Desse modo, caso desejemos visualizar uma ajuda rápida sobre um comando externo, devemos fazer da seguinte forma:

```
 # [comando] --help
```

O parâmetro --help pode ser utilizado em qualquer comando para ter uma consulta rápida dos parâmetros que determinado comando pode nos oferecer. É importante entender que --help é na verdade um parâmetro individual de cada comando, logo se um comando não tiver esse parâmetro existem outros meios para se obter ajuda.

Não se esqueça de estudar as diferenças entre comandos internos e externo

6.4.2. Comando man

O comando man é o responsável por trazer os manuais mais completos sobre determinado comando, arquivo de configuração, bibliotecas, entre outros nos quais estamos trabalhando.

Os manuais do sistema são divididos em níveis que são os seguintes:

- man 1 - Programas executáveis e comandos do Shell;
- man 2 - Chamadas de sistema (funções providas pelo Kernel);
- man 3 - Chamadas de bibliotecas (funções como bibliotecas do sistema);
- man 4 - Arquivos de dispositivo (Localizados normalmente no /dev);

- man 5 - Arquivos de configuração e convenções;
- man 6 - Jogos;
- man 7 - Variados (incluindo pacotes de macros e convenções);
- man 8 - Comandos de administração do sistema (normalmente usado somente pelo root);
- man 9 - Rotinas de Kernel.

É comum o exame cobrar mais pelos níveis 1, 5 e 8 dos manuais!

Sintaxe do comando man:

```
# man [comando]
```

ou

```
# man [seção] [comando]
```

Observação: Essas informações sobre as seções do comando man podem ser achadas no seu próprio manual digitando o comando man man.

Caso desejemos visualizar o manual do comando passwd, podemos fazer da seguinte forma:

```
# man passwd
```

Estamos consultando o manual do comando passwd. Para navegar pelo manual, o comando man abre um arquivo que está compactado na pasta /usr/share/man/man1 para o passwd ou outro nível de manual dependendo do comando ou arquivo.

O passwd é conhecido no sistema GNU/Linux como um comando que adiciona ou modifica a senha do usuário e como o arquivo de usuários do sistema (/etc/passwd). Veremos agora o manual do arquivo de usuários passwd:

```
# man 5 passwd
```

Podemos consultar quais manuais estão disponíveis dentro do próprio diretório do man:

```
 # /usr/share/man/
```

Dentro desse diretório é possível ver todas as divisões dos manuais dentre elas os níveis e as línguas.

Todos os níveis de manuais possuem sua determinada introdução que pode ser vista com o comando:

```
# man <nível> intro
```

Podemos ver os manuais em diversas línguas diferentes, desde o pacote para a língua escolhida esteja instalado. Se nosso sistema estiver instalado em português, o comando man irá trazer todas os manuais disponíveis em português. Já se nosso sistema estiver em inglês é preciso usar o parâmetro **-L pt_BR**, para que possamos ver os manuais na nossa língua:

```
# man -L pt_BR comando
```

É importante nesse ponto ressaltar que a documentação em nossa linguagem depende de pessoas que ajudam a fazer a tradução para o português, se você quiser ajudar, acredite, você será muito bem vindo, veja como ajudar com o comando:

```
# man 7 undocumented
```

Podemos ver que para visualizar o manual do arquivo de usuário passwd precisamos informar em qual nível de manual ele se encontra, pois já existe um passwd no nível 1 que é o comando, então ele aparece primeiro quando digitamos man passwd. Esse manual do arquivo passwd está compactado na pasta **/usr/share/man/man5.**

6.4.3. Comando apropos

O comando apropos é utilizado quando não se sabe qual documentação acessar para um determinado assunto, mostrando as man pages que contém a palavra-chave que foi especificada.

A sintaxe utilizada para usar o apropos é a seguinte:

```
# apropos [palavra-chave]
```

Para localizar as man pages, o comando apropos utiliza um banco de dados construído com o comando catman (executado pelo administrador do sistema, root).

Uma forma equivalente ao apropos é o comando man juntamente com a opção -k:

```
# man -k [palavra-chave]
```

Para construir o banco de dados do comando apropos deveremos fazer da seguinte forma:

```
# catman
```

Na Red Hat, o comando catman foi substituído pelo makewhatis.

Os comandos apropos e whatis dividem a mesma base de dados, é importante perceber isso.

6.4.4. Comando whatis

O comando whatis tem basicamente a mesma função do comando apropos, só que as buscas do comando whatis são mais específicas. O apropos busca as páginas de manuais e descrições de maneira mais genérica; se digitarmos a palavra passwd ele nos trará tudo que tiver passwd, seja como nome ou parte do nome do manual ou na descrição. E o whatis nos trará somente o manual com nome exato da palavra pesquisada.

A sintaxe utilizada no comando whatis é a seguinte:

```
# whatis [comando]
```

Uma forma equivalente ao whatis é o comando man juntamente com a opção -f:

```
# man -f [palavra-chave]
```

6.4.5. Comando info

As info pages são como as páginas de manuais, porém são utilizadas com navegação entre as páginas. Elas são acessadas pelo comando info. O comando info é útil quando já sabemos o nome do comando e só queremos saber qual sua respectiva função.

A navegação das info pages é feita através de nomes marcados com um ``*'' (hipertextos) que, se pressionarmos **Enter**, nos levará até a seção correspondente, e backspace volta à página anterior.

Podemos também navegar pelas páginas com as teclas **n (next/próximo); p (previous/anterior); u (up/sobe um nível).**

Para sair do comando info, basta teclar **q.**

Caso desejemos exibir a lista de todos os manuais de comandos/programas disponíveis, podemos fazer da seguinte forma:

```
# info
```

Para exibir as informações somente de um determinado comando, usaremos a seguinte sintaxe:

```
# info [comando]
```

6.5. Alternativas para consulta

Para obtermos um melhor forma de visualização, duas ferramentas de documentação foram desenvolvidas:

- **yelp** - Ferramenta gráfica para visualização de manuais de aplicativos gráficos do GNOME;
- **xman -** Front-end para man, assim facilitando a consulta das manpages;
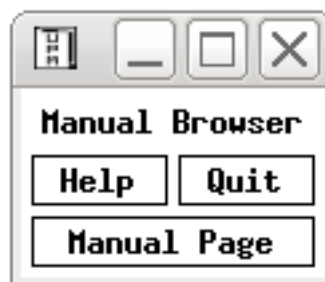
Ilustração 3: xman
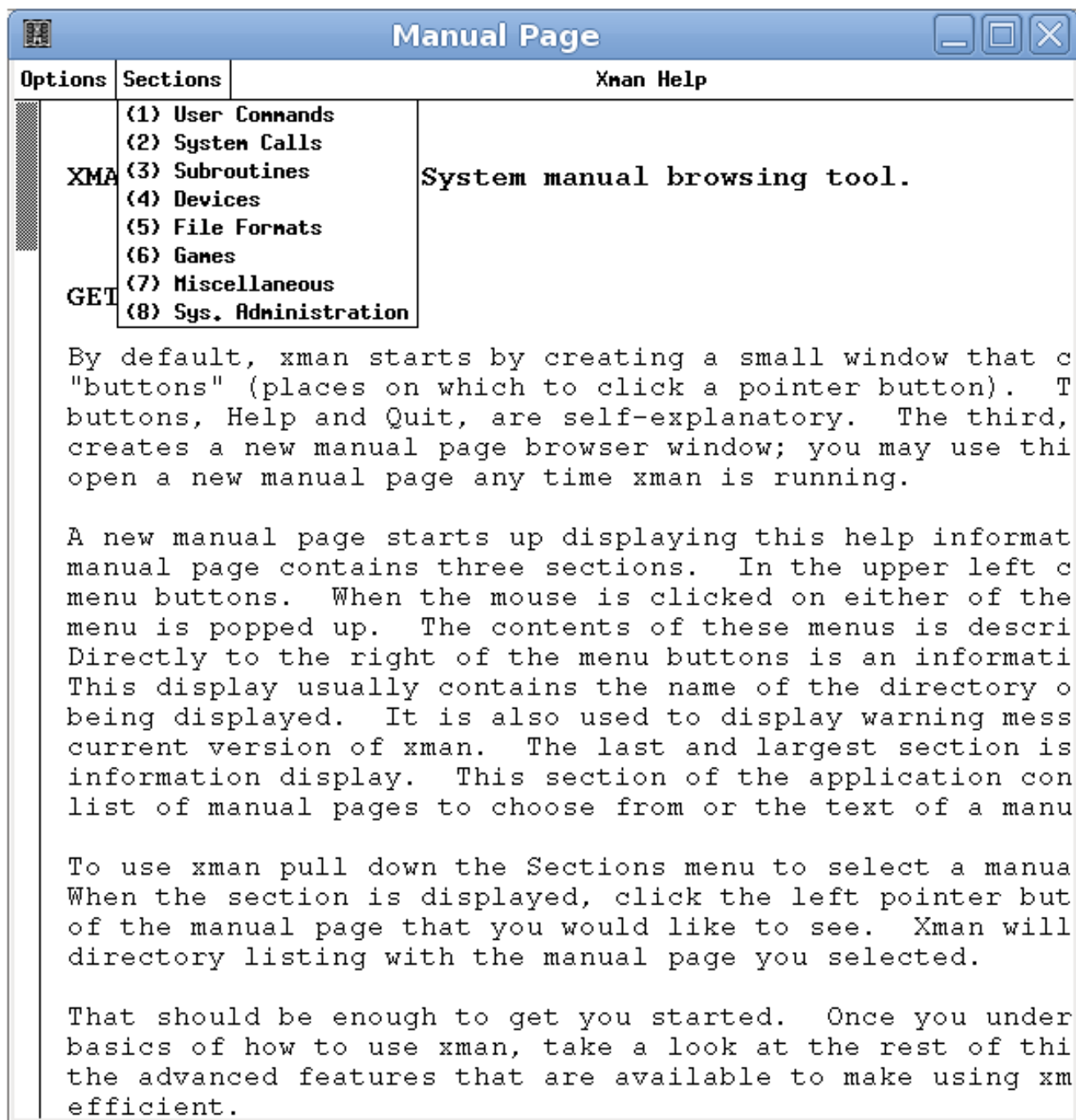
Ilustração 4: Menu do xman

6.6. Comando whereis

O comando whereis é utilizado para mostrar a localização do binário do comando, do arquivo de configuração (caso exista) e a localização das páginas de manuais do determinado comando ou arquivo.

Para visualizarmos a localização destes dados para um determinado comando ou arquivo, utilizamos a seguinte sintaxe:

```
# whereis [comando]
```

ou

```
# whereis [arquivo]
```

6.7. Comando which

O comando which é bem semelhante ao comando whereis, só que o comando which só mostra a localização do binário do comando.

Para visualizar a localização do binário do comando, utilizamos a seguinte sintaxe:

```
# which [comando]
```

O comando which é muito usado para abreviações de comandos em shell script. Podemos testar executando o comando abaixo:

```
$ lsetc=”$(which ls) -la /etc --color”
$lsetc
```

6.8. Prática Dirigida

Visualize uma ajuda rápida para o comando interno cd:

```
# help cd
```

Abra o manual do comando man:

```
# man man
```

Abra o manual do comando ifconfig e veja em qual nível de manual ele se encaixa:

```
# man ifconfig
```

Qual path (localização) completo do comando e arquivo passwd e respectivas páginas de manual?

```
# whereis passwd
```

Qual o path (localização) somente do binário do comando ls?

```
# which ls
# which source
```

Como visualizamos as páginas de informação de todos os comandos?

```
# info
```

Busque os manuais que possuem a palavra user em suas descrições:

```
# apropos user
```

Busque os manuais que tenham somente o nome passwd:

```
# whatis passwd
```

Onde podemos encontrar no sistema uma documentação e um arquivo de exemplo da aplicação apt?

```
# cd /usr/share/doc/apt
```

Visualize alguns arquivos de howto do firewall:

```
# cd /usr/share/doc/iptables/html/
# w3m NAT-HOWTO.html
```

6.9. Exercícios Teóricos

1) Quantos são os níveis de manuais existentes? Como eu vejo uma descrição de cada um deles?

____

____

____

____

2) Qual é o site oficial de documentação do sistema GNU/Linux?

____

____

____

____

3) Como devemos proceder caso não saibamos o nome do comando específico que desejamos consultar o manual? Por exemplo, qual comando em modo texto utilizaríamos para enviar um e-mail?

____

____

____

____

4) Em qual nível de manual se encaixa o comando ifconfig?

____

____

____

____

5) Qual é o nível de manual para arquivos de configuração?

____
____
____
____

6) Qual é a diferença entre how-to, manual e documentação?

____
____
____
____

7) Em qual nível do man se encontra a biblioteca DATE::FORMAT?

____
____
____
____

6.10. Laboratório

1) Gere a base de manuais do sistema.
2) Pesquise em qual nível de manual se encaixa o shadow.
3) Descubra qual é a localização do comando su.
4) Descubra se source é um comando interno ou externo.
5) Tente descobrir se existe no sistema algum manual ou documentação que fale sobre a hierarquia do sistema de arquivos.

Capítulo 7

Editores de texto

7.1. Objetivos

- Conhecer os diversos editores textos;
- Explorar o editor de textos nano;
- Explorar o editor de textos Vim;

7.2. Introdução

A grande maioria das configurações em sistemas GNU/Linux são feitas editando-se diretamente arquivos de configuração em modo texto. Para que essa tarefa seja executada com sucesso, é preciso conhecer alguns editores do texto dentre eles: vi, vim, nano, pico, mcedit, ed, emacs dentre muitos outros:

- **vi -** Sem dúvida nenhuma o editor mais famoso de todos os tempos, presente em quase todas as distribuições;

- **vim -** Uma versão melhorada do vi, Vim significa VImpruved e trás diversas facilidades sem perder os conceitos do vim;
- **nano -** Editor padrão de muitas distribuições como Debian e Ubuntu, esse editor diferente do vim é muito fácil de ser usado;
- **pico -** Muito parecido com o nano, este está presente nas distribuições Slackware e Gentoo;
- **mcedit -** Editor muito fácil e completo que está presente nas distribuições Red-Hat, CentOs, Seu grande diferencial é a possibilidade da utilização do mouse;
- **ed -** O editor mais simples de texto presente no mundo Unix, o ed é um editor de linha para terminais aonde não é possível abrir uma janela de edição;
- **emacs -** Poderoso editor de "tudo", o emacs também é muito conhecido no mundo GNU/LINUX por fazer muitas coisas diferenciadas de um editor de texto;

*O editor **od**, pode ser importante para prova. Sua principal funcionalidade é mostrar a saida de um arquivo em formatos como octal, ASCII entre outros.*

Nesse capitulo vamos abordar apenas a utilização dos editores nano e vim.

7.3. Editor Nano

O nano é o editor padrão de textos do Debian, e distribuições baseadas nele. Esse editor é muito fácil de ser usado, e sua interface é muito intuitiva e agradável.

Para abrirmos o editor devemos chamar o seguinte comando:

```
$ nano
```

```
^G Ajuda      ^O Gravar      ^R Ler o Arquivo  ^Y Página Anterior  ^K Recortar Texto  ^C Pos Atual
^X Sair       ^J Justificar  ^W Onde está?     ^V Próxima Página   ^U Colar Txt       ^T Para Spell
```

Ilustração 5: Menu do Nano

Ao ser chamado, este editor irá apresentar um tela em branco com um rodapé semelhante a esse:

Vamos analisar essas funções:

Lembrando que ^G é igual a Ctrl + G e assim por diante ...

- **^G Get Help** - Apresenta uma tela de ajuda os mais diversos comandos e uma breve explicação sobre o editor;
- **^X Exit -** Sai do editor, lembrando que se o arquivo não estiver salvo, essa opção irá te pedir para salvar;
- **^O WriteOut -** Salva ou sobrescreve um arquivo;
- **^J Justify -** Justifica o arquivo inteiro;
- **^R Read File** - Abre um arquivo;
- **^W Where Is** - Procura por uma ocorrência dentro do arquivo;
- **^Y Prev Page** - Move o cursor para pagina anterior;
- **^V Next Page** - Move o cursor para próxima pagina;
- **^K Cut Text** - Corta a linha em que o cursor está posicionado;
- **^U UnCut Text** - Cola a linha recortada na posição atual do cursor
- **^C Cur Pos** - Mostra informações sobre a posição do cursor;
- **^T To Spell** - Auto correção, lembrando que é necessário ter o comando spell instalado;

Como podemos ver usar o editor de textos nano, não é uma das tarefas mais difíceis no GNU/Linux. Vamos ver agora o editor Vim

As variáveis relacionadas com os editores podem ser valiosas na prova. Uma delas é a própria variável EDITOR

```
EDITOR=nano visudo
EDITOR=vim visudo
```

Observem que a variável define o editor que abrirá um programa que chama o editor padrão. Para definirmos quem é o editor padrão podemos usar o aplicativo update-alternatives .

```
 update-alternatives --config editor
```

7.4. Editor Vim

O Vi é o editor básico do GNU/Linux, está disponível em grande parte das distribuições do GNU/Linux , mesmo naquelas que vêm em apenas um disquete.

Hoje em dia, as distribuições usam uma versão mais completa e com mais recursos do que o Vi que é o Vim (VI iMproved). Abaixo podemos ver uma tela do editor de textos vim:

```
~
~
~
~
~
~
~
~
~                         VIM - Vi IMproved
~
~                          version 7.1.314
~                     by Bram Moolenaar et al.
~            Vim is open source and freely distributable
~
~                    Help poor children in Uganda!
~          type  :help iccf<Enter>       for information
~
~          type  :q<Enter>               to exit
~          type  :help<Enter>  or  <F1>  for on-line help
~          type  :help version7<Enter>   for version info
~
~
~
~
~
~
~
```

Ilustração 6: Tela principal do vim

Ao invocar o Vi, este entra direto para o modo ``visualização''. Para modificar o arquivo, usam-se os modos de inserção, deleção e de substituição. Para voltar ao modo de visualização, sempre se usa a tecla ESC.

Ilustração 7: Fluxo de Trabalho do vim

Todos os serviços em unix são configurados através de arquivos de configuração, o vim não seria diferente. Seu arquivo de configuração fica localizado em /etc/vim/vimrc. Para configurar o seu editor de textos, basta descomentar as funcionalidades desejadas, e copiar o arquivo para o seu home como **.vimrc**

```
cp /etc/vim/vimrc ~/.vimrc
```

7.5. Prática Dirigida

1) Faça uma cópia de segurança do arquivo /etc/passwd para seu diretório home:

```
cp /etc/passwd ~
```

2) Vamos testar alguns comandos básicos do vim:

```
vim passwd
```

7.5.1. Teste os comandos de Edição

3) Comandos básicos de inserção de texto:

- **i -** Insere texto antes do cursor
- **a -** Insere texto depois do cursor
- **r -** Substitui texto no início da linha onde se encontra o cursor
- **A -** Insere texto no final da linha onde se encontra o cursor
- **o -** Adiciona linha abaixo da linha atual
- **O -** Adiciona linha acima da linha atual
- **Ctrl + h -** Apaga o último caractere

Teste os comandos de Movimentação

4) Comandos básicos de movimentação:

- **Ctrl+f -** Move o cursor para a próxima tela
- **Ctrl+b -** Move o cursor para a tela anterior
- **H -** Move o cursor para a primeira linha da tela

- **M -** Move o cursor para o meio da tela
- **L -** Move o cursor para a última linha da tela
- **h -** Move o cursor um caractere à esquerda
- **j -** Move o cursor para a próxima linha
- **k -** Move o cursor para linha anterior
- **l -** Move o cursor um caractere à direita
- **w -** Move o cursor para o início da próxima palavra
- **W -** Move o cursor para o início da próxima palavra, separadas por espaço
- **b -** Move o cursor para o início da palavra anterior
- **B -** Move o cursor para o início da palavra anterior, separadas por espaço
- **0(zero) -** Move o cursor para o início da linha atual
- **^ -** Move o cursor para o primeiro caractere não branco da linha atual
- **$ -** Move o cursor para o final da linha atual
- **nG -** Move o cursor para a linha n
- **:n -** Move o cursor para a linha n
- **gg -** Move o cursor para a primeira linha do arquivo
- **G -** Move o cursor para a última linha do arquivo

Escreve um texto e teste comandos de Localização

5) Comandos básicos para localizar texto:

- **/palavra -** Busca pela palavra ou caractere em todo o texto
- **?palavra -** Move o cursor para a ocorrência anterior da palavra
- **n-** Repete o último comando / ou ?
- **N -** Repete o último comando / ou ?, na direção reversa
- **Ctrl+g -** Mostra o nome do arquivo, o número da linha atual e o total de linhas

Teste também os comandos de Alteração

6) Comandos básicos para alteração de texto:

- **x -** Deleta o caractere que está sob o cursor ;

- **dw -** Deleta a palavra, da posição atual do cursor até o final ;
- **dd -** Deleta a linha atual, e copia o conteúdo para área de transferência ;
- **D -** Deleta a linha a partir da posição atual do cursor até o final ;
- **:A,Bd -** Deleta da linha A até a linha B, copia para área de transferência ;
- **rx -** Substitui o caractere sob o cursor pelo especificado em x ;
- **u -** Desfaz a última modificação ;
- **U -** Desfaz todas as modificações feitas na linha atual ;
- **J -** Une a linha corrente a próxima ;
- **yy -** Copia 1 linha para a área de transferência ;
- **yNy -** Copia N linhas para a área de transferência ;
- **p -** Cola o conteúdo da área de transferência ;
- **Np -** Cola N vezes o conteúdo da área de transferência ;
- **cc -** Apaga o conteúdo da linha, e copia para área de transferência ;
- **cNc -** Apaga o conteúdo de N linhas, e copia para área de transferência ;
- **:%s/string1/string2/g -** Substitui "string1" por "string2" ;

Por fim teste os comandos de Execução

7) Comandos para salvar o texto:

- **:wq ou :x -** Salvam o arquivo e saem do editor
- **:w nome_do_arquivo -** Salva o arquivo corrente com o nome especificado
- **:w! nome_do_arquivo -** O mesmo que :w, mas forçando sobrescrita
- **:q -** Sai do editor
- **:q! -** Sai do editor sem salvar as alterações realizadas

Algumas dicas de vi para a lpi:

:setnumber

:syntax on

:noai

:set hlsearch

:set background=dark

7.6. Exercício Teórico

1) Como removo as linhas 40 a 60?

2) Como vou para o início do arquivo?

3) Como apago as 8000 primeiras linhas ?

4) Como vou para a linha 25 do arquivo?

5) Qual a sequencia de teclas que apaga a linha atual e mais 3 linhas abaixo do cursor?

6) Como faço uma cópia do meu arquivo para a seguinte path: /tmp/backup?

7) Como eu copio 5 linhas, e depois colo?

8) Como eu acho todas as palavras ``linux'' dentro do texto?

____

____

____

____

9) Um arquivo tem 620 linhas, eu quero ir para a última linha do arquivo.

____

____

____

____

10) Substituir todos os /, por @ dentro do arquivo?

____

____

____

____

7.7. Laboratório

1) Crie um arquivo chamado dados_pessoais.
2) Dentro dele, coloque um dado em cada linha: Nome completo, e-mail e um site de sua preferência. Grave o arquivo.
3) Agora, modifique algum dado e tente sair do editor sem salvar.
4) De dentro do Vi, faça uma cópia desse arquivo para dados_pessoais2.
5) Faça uma cópia da sua linha de e-mail 5 vezes.
6) Posicione o cursor na primeira linha do arquivo, sem usar as teclas direcionais.
7) Vá direto para a linha 3, sem usar as teclas direcionais.
8) Vá direto para o final do arquivo, sem usar as teclas direcionais.
9) Recorte a linha do seu nome, coloque-a no final do arquivo.
10) Apague a linha que tem seu nome e depois tente desfazer a sua última ação.
11) Faça uma cópia do arquivo /etc/inittab para o seu HOME. (cp /etc/inittab ~)

12) Ainda no arquivo, entre no modo visual.

13) Apague as 5 primeiras linhas do arquivo.

14) Substitua todas as string init (minúsculo) por INIT (maiúsculo).

15) Localize a string ``shutdown'', usando o método de busca.

16) Substituir das linhas 22 a 28 a string ``wait'' pela string ``esperar''.

17) Dentro do arquivo, localize todas as palavras ``respawn'' e apague todas elas.

18) Apague as linhas 19 e 24.

19) Mova as linhas 35,36,37 e 38 para o final do arquivo.

16) Salvar o arquivo utilizando um novo nome e salve-o no diretório /root.

Capítulo 8

Introdução a Redes

8.1. Objetivos

- Introdução a Teórica de redes TCP/IP;
- Conceitos básicos de configurações de redes em UNIX;
- A importância de alguns elementos das redes de computadores;

Neste capítulo, iremos aprender alguns conceitos de redes que são muito importantes no nosso dia a dia em TI. Elementos como o IP da máquina e a máscara de rede, são de fundamental importância quando falamos de uma configuração de rede. Tão importante quando os itens acima iremos aprender o quão importante é saber como funciona uma rede, sabendo configurar seu gateway e definir seu DNS, alem de descobrir algumas coisas que facilitam a configurações diária de redes nos nossos sistemas UNIX.

8.2. Os Protocolos TCP/IP

Os protocolos TCP/IP antigamente eram usados como um padrão militar para troca de informações. Atualmente esses protocolos são os padrões mundiais para comunicação de redes.

O protocolo TCP(Transmission Control Protocol), é orientado a conexões, transporta informações por meio de handshaking, ou por meio de retransmissão caso algum erro aconteça. Esse protocolo garante o envio das mensagens. Podemos citar alguns serviços de rede que utilizam o protocolo TCP: SMTP, FTP e Telnet

Já o protocolo IP(Internet Protocol) descrito pela RFC 791, é responsável por estabelecer o esquema de endereçamento e pela definição de datagramas.

Apesar da nova verão da LPI ter diminuido os tópicos de TCP/IP, ele ainda está presente na prova, e por isso olhar um pouco o modelo OSI é uma boa idéia.

Acompanhe um pouco mais desse assunto no treinamento 451 da Formação 4Linux.

8.3. Entendendo o IP

O endereçamento IP, como dever ser chamado, é composto por 4 octetos e uma máscara, que determina quantos endereços são destinados a hoste e quantos endereços são destinados a rede.

No mundo GNU/Linux não diferente dos outros, para termos acesso a internet ou a comunicação em rede também precisamos ter nosso numero IP. O numero IP está presente em todas as máquinas, mesmo nas que não tem conexão com a internet.

Isso é possível pois em todo GNU/Linux há uma interface lógica, chamada loopback (lo) cujo endereço IP será 127.0.0.1 e que sempre deve estar devidamente configurada.

Nosso endereço de loopback atende também por 0.0.0.

Você pode estar se perguntando:
Mas por que raios eu poderia querer um serviço em uma máquina que não fala com o mundo externo?".

A resposta é simples, você pode desenvolver suas páginas e sistemas Web e testá-las localmente. Ou mesmo testar a implantação de um servidor de DNS ou um proxy. Antes mesmo de colocar ele em produção e fazer com que seus queridos usuários reclamem de algo que não funcionou direito. Em resumo, você pode fazer qualquer coisa que você queira e não necessariamente precisa ter contato com o mundo exterior.

Com essa interface configurada, todo o tipo de serviço pode ser ativado na máquina, desde um simples servidor de ssh até um servidor de DNS passando por um servidor de páginas de Web afim de realizar testes antes mesmo de colocar esses serviços em produção.

O mundo exterior, a Internet é toda feita por números IP's, e não depende, isso mesmo, não depende em momento nenhum de um servidor de DNS para funcionar, esse serviço de DNS somente facilita o nosso acesso a internet, permitindo nosso acesso a ela através de nomes e não de números, isso significa que para entenderemos como a internet funciona, precisamos entender como funcionam os números que ela utiliza, os números IP's.

Para configurarmos um número IP em nosso computadores, precisamos também, configurar um Máscara para esse numero IP. A mascara de rede, também conhecida como netmask, é um numero constituído por 32 bits, que é utilizado para separar redes, determinando quem é host, quem é rede e quem é broadcast.

- **Host -** Um endereço disponibilizado para computadores poderem acessar a rede;
- **Rede -** Um endereço da rede, que com a ajuda da máscara delimita qual é o começo e o fim da rede;
- **BroadCast -** Normalmente é o ultimo endereço da rede, utilizado para que

uma máquina possa falar com todas as outras.

Ilustração 8: Entendendo as Mascaras

Sabendo como funciona a máscara de rede é necessário saber que existem três classes padrões de Redes

Classe A 11111111.00000000.00000000.00000000

Classe B 11111111.11111111.00000000.00000000

Classe C 11111111.11111111.11111111.00000000

Rede Host

Ilustração 9: Conhecendo as classes

Ainda compreendendo esses números, precisamos descobrir que existem dois tipos de endereçamento IP. Endereços Públicos e Privados

- **Ip Público -** São endereços válidos para internet, rede de comunicações.
- **Ip Privado -** São endereços inválidos para internet, podendo somente ser

usados em redes privadas como Lan's ou WLans.

Ilustração 10: Ip's públicos e privados

Para entendermos melhor esse conceito é preciso entender qual é o papel do NAT nas redes de computadores. O NAT(Network Address Translation) é uma técnica desenvolvida devido aos números IP's (IPv4) estarem se esgotando rapidamente.

Na história do endereçamento IP, tivemos alguns abalos que nos fizeram quase usar o Ipv6 antes da hora. Antes do NAT existir, não existia o conceito de IP's publicos e privados, assim os IP's estavam se esgotando de forma muito rápida. E com a técnica de NAT é possível fazer um melhor aproveitamento os números IP's para dar um maior tempo de vida para o protocolo IPv4, assim então utilizando o conceito de IP Privado para economizar a quantidade de Ip's Públicos.
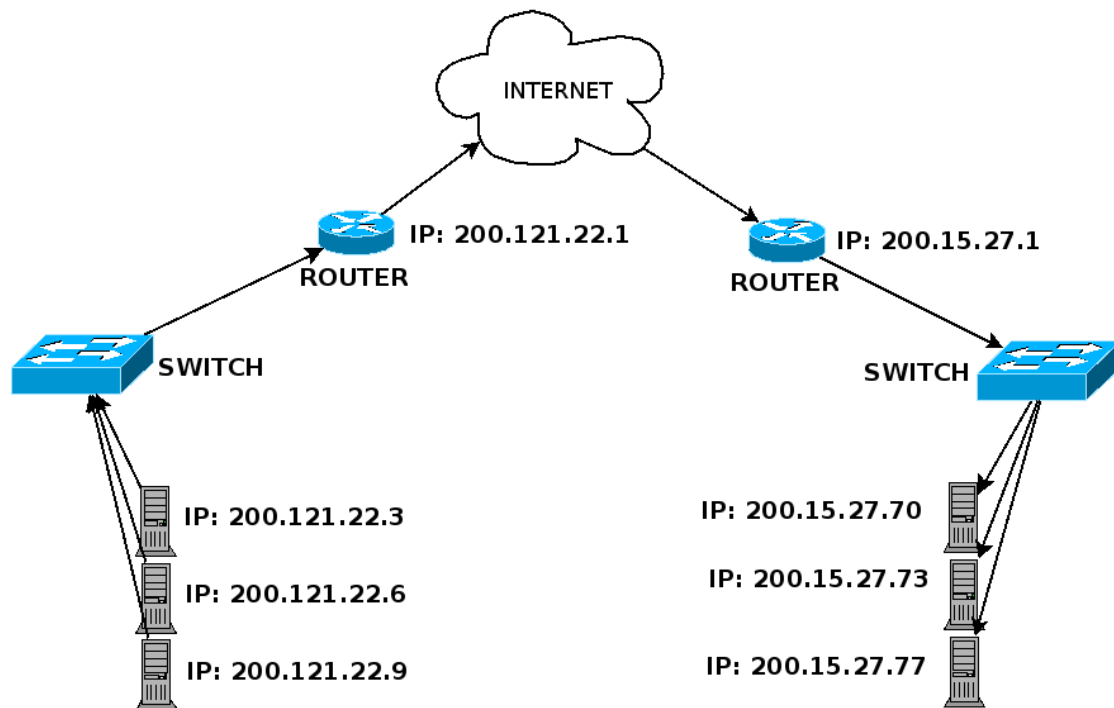
Ilustração 11: Antes do NAT

Se cada um de nossos computadores utiliza-se um numero IP Publico, a quantidade desses IP's já teria se esgotado, e provavelmente já estaríamos utilizando o protocolo IPv6. Vamos descobrir agora os ranges de IP's que podemos utilizar, os ips Privados.
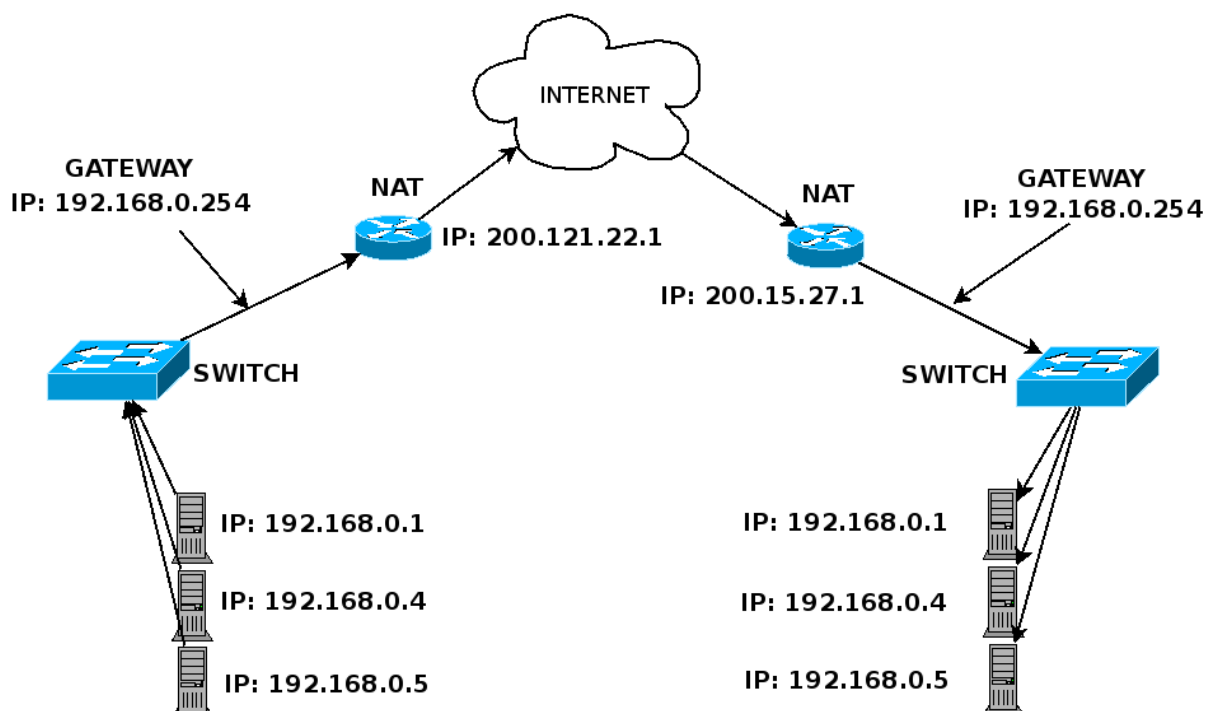
Ilustração 12: Depois de um NAT

8.4. Entendendo o gateway da rede

O gateway da rede é um host que conhece outros host que por sua vez conhecem outros. Complicado?

O principal papel do gateway é levar os pacotes tcp/ip para outras redes quais os hosts que os originaram não conhecem. É dessa forma que os pacotes saem de uma rede privada para um rede Wan. Para que os pacotes possam transitar pela internet ou mesmo só por uma rede fechada é necessário um gateway. No caso de vocês estar em sua própria rede local, o seu gateway é você mesmo, pois todos os hosts estão normalmente com a mesma configuração de IP, ou seja, mesma mascara, mesma classe de IP, e etc.

8.5. O servidor DNS

Nesse ponto é muito importante frisar que o servidor de DS não faz parte da configuração essencial de rede, pois para estarmos na internet, basta termos um gateway que a conheça configurado. Lembre-se a internet é feita de números:

Usuário - *Suporte?*

Suporte - *Sim, em que posso ajudar?*

Usuário - *A internet está fora do ar ... (O cara não consegue acessar o host orkut.com =( )*

Para resolver esse probleminha basta digitar o comando:

ping 4.2.2.2

Se a resposta for positiva, você não tem um problema de link, cheque seu DNS.

Suporte - *Sr. Usuário, percebi que você está acessando um site proibido pela empresa, há algo errado?*

Usuário - *Eu?? .. Não, não, tudo bem, a internet já está normalizada.*

8.6. Arp e RARP

Vamos nos aprofundar um pouco mais nas teorias de redes e vamos dar uma olhadinha nos protocolos ARP e RARP. O protocolo ARP é utilizado para converter os endereços de rede (IP's), para os endereços físicos das interfaces (MAC). Um exemplo clássico de usabilidade no Brasil é: Placas com o mesmo mac address na rede. Podemos conhecer todas as máquinas da rede e depois utilizar o arp para descobrir qual dos ips tem o mesmo Mac address igual. Já o Rarp faz exatamente o oposto, trasforma Endereços físico em endereços de Rede.

8.7. Configurando a Rede

A configuração de rede em um sistema GNU/Linux é muito importante pois esses sistemas são, intrinsecamente, sistemas de rede. Ou seja, mesmo que não haja nenhum tipo de interface de rede ou modem ou qualquer outra coisa do gênero, ainda assim uma máquina GNU/Linux será um sistema de rede.

A configuração da rede se base-a em três etapas:

- Configuração do numero IP e máscara
- Configuração do Gateway
- Configuração dos DNS Servers

8.7.1. Configurando IP e Máscara

Além da interface lo podemos configurar outras interfaces, basta que elas estejam presentes e sejam suportadas pelo kernel. Na maior parte dos casos, a interface mais comum acaba sendo a interface **eth0** de ethernet. Para configurar a nossas interfaces de redes utilizamos o comando **ifconfig.**

```
# ifconfig
```

A utilização do comando ifconfig é um item essencial para o assunto de redes. Outro comando que pode der um bom valor na prova é o comando iwconfig que veremos no treinamento 451 da formação 4Linux

Com esse comando é possível descobrir todas as interfaces presentes no sistema, mas para ter certeza que nenhuma interface está inativa adicionamos o parâmetro **-a.**

```
# ifconfig -a
```

Para atribuir um endereço IP para uma placa de rede utilizamos essa sintaxe:

```
# ifconfig <interface> <IP>
```

Exemplo:

```
# ifconfig eth0 192.168.32.54
```

Como esse comando estamos atribuindo o endereço IP 192.168.32.54 para a interface **eth0.**

O comando ifconfig calcula automaticamente a máscara, mas se você precisar configurar uma mascara diferenciada, você deve usar o parâmetro netmask:

```
# ifconfig eth0 192.168.32.54 netmask 255.255.254.0
```

Para ativar ou desabilitar um placa de rede podemos usar a sintaxe:

```
# ifconfig eth0 up
# ifconfig eth0 down
```

*Uma boa alternativa para habilitar e desabilitar as placas de redes, seriam os comando **ifup e ifdown.** Ainda avançando na configuração da rede, o próximo passo é a configuração do gateway da rede.*

8.7.2. Configurando o gateway

Para que nossos pacotes saibam para onde ir eles precisam conhecer o IP do Gateway da rede. O papel do gateway da rede é simples, ele funciona como uma saída para todos os pacotes daquela rede, para outras redes.

Para configurar o gateway da nossa rede utilizamos o comando route com os seguinte parâmetros:

```
# route add default gw IP
```

Com esse comando é possível configurar a rota padrão de saída da nossa rede. Para listar todas as rotas traçadas, podemos utilizar o comando abaixo:

```
# route -n
```

Com ele podemos descobrir se as rotas necessárias para que nossa rede funcione estão corretas.

Se desejarmos remover a rota padrão, devemos utilizar o comando:

```
# route del default
```

Esse comando se encarregará de remover a rota padrão para a saída da rede, mas lembre que essa rota é obrigatória no processo de configuração de rede.

8.7.3. Configuração dos DNS Servers

Para não termos que decorar todos os números Ip's que precisamos acessar foi criado um serviço de rede chamado DNS. O DNS faz a tradução de nomes para números IPs e vice-versa. Esse serviço de rede será melhor detalhado no curso 452 da Formação 4Linux.

Para configurar os servidores de DNS para máquina local, precisamos editar o arquivo de configurações de DNS, chamado resolv.conf localizado em /etc.

```
# vim /etc/resolv.conf
```

Dentro deste arquivo podemos configurar nossos servidores de DNS, coloque nesse arquivo as seguintes linhas:

```
nameserver 201.6.0.100
```

Com essa sintaxe acabamos de configurar um servidor de DNS, no caso o DNS do Virtua.

Os comandos traceroute e tracepath podem ajudar nos administradores a descobrir em que ponto da rede podemos ter um possível problema .

8.7.4. Configuração estática de rede

Tudo que vimos até agora, são configurações que podem ser atribuídas através de linha de comando (configurações dinâmicas). Porém nosso host deve estar devidamente configurado para que, por exemplo, após um boot, a máquina esteja com as configurações certas.

Para que isso aconteça temos que configurar o arquivo **/etc/network/interfaces:**

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
 address 192.168.200.X
 netmask 255.255.255.0
 broadcast 192.168.200.255
 network 192.168.200.0
 gateway 192.168.200.254
```

Os arquivos de configuração das interfaces de rede estão localizadas em /etc/sysconfig/networking/devices. A sintaxe é diferente e recomenda-se utilizar o aplicativo netconfig para configurar a rede.

8.8. Arquivo Hosts

Podemos também configurar alguns atalhos para alguns endereços de rede. Esses atalhos ficam localizados dentro do arquivos /etc/hosts.

A sintaxe dele é:

```
 IP      apelido   apelido
```

Exemplo :

```
192.168.200.254          instrutor professor
```

Isso facilita nosso trabalho, uma vez que todos estão devidamente apelidados, não precisamos mais decorar números IP.

8.9. Comando hostname

O comando hostname altera dinamicamente o nome da máquina e deve ser utilizado da seguinte maneira:

```
#  hostname NOVONOME
```

Para alterar o hostname de maneira estática, devemos editar o arquivo **/etc/hostname:**

O comando hostname com sua opção -f (FQDN) pode ser muito útil para seu dia-a-dia!!!

Um exemplo de configuração de fqdn seria colocar a linha abaixo no arquivo **/etc/host .**

```
 IP FQDN HOSTNAME
200.12.44.211  zeus.criptahacks.ntr.au   zeus
```

8.10. O arquivo nsswitch.conf

Presente em /etc o arquivo nsswitch.conf, nos permite configurar qual será a ordem de busca por logins válidos na estação, ou seja, se a máquina em questão precisar buscar o login em um servidor ldap, nis ou outro meio de autenticação, é nesse arquio que devemos especificar essa configuração.

```
passwd:compat
group: compat
shadow:compat
```

Acima temos a configuração padrão para buscar nos arquivos de senha do sistema. Mas se precisarmos autenticar em um servidor ldap a configuração ficaria assim:

```
passwd:files ldad
group: files ldap
shadow:files ldap
```

Nessa configuração temos o temo files, dizendo qual é o sobre que meio ocorrerá essa autenticação. Outros valores podem ser aplicados como db e nis.

8.11. Prática Dirigida

1) Verifique as configurações de rede que estão ativas:

```
# ifconfig
```

2) Verifique que rotas estão sendo utilizadas:

```
# route -n
```

3) Determine quais as interfaces de rede estão disponíveis:

```
# ifconfig -a
```

4) Configure a interface eth0 manualmente para que utilize um IP da rede 192.168.200.X com máscara de sub-rede 255.255.255.0, sendo X o número da sua máquina:

```
# ifconfig eth0 192.168.200.X
```

5) Verifique as configurações de rotas:

```
# route -n
```

6) Adicione uma rota para o gateway 192.168.200.254:

```
 # route add default gw 192.168.200.254
# route -n
```

7) Envie um icmp tipo echo-request (ping) para uma outra máquina da rede:

```
 # ping 192.168.200.Y
```

8) Se possível, pingue todas as máquinas da rede utilizando o endereço de broadcast:

```
 # ping -b 192.168.200.255
```

9) Determine os endereços MAC das interfaces que responderam ao ping:

```
 # arp -n
```

10) Desative a interface de rede e ative-a novamente:

```
 # ifconfig eth0 down
# ifconfig
# ifconfig eth0 up
# ifconfig
```

11) Vamos configurar nossas interfaces de rede editando o arquivo ***/etc/network/interfaces:***

```
 # vim /etc/network/interfaces
```

12) Altere o conteúdo dele para satisfazer as configurações da sua rede:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
 address 192.168.200.X
 netmask 255.255.255.0
 broadcast 192.168.200.255
 network 192.168.200.0
 gateway 192.168.200.254
```

13) Pare o serviço de rede e reinicie-o na seqüência:

```
# invoke-rc.d networking stop
# invoke-rc.d networking start
# ifconfig
```

Para iniciar serviços no Red Hat, recomenda-se utilizar o comando service. Para iniciar/parar/reiniciar a interface de rede, use o comando service network start/stop/restart.

14) Efetue pingues para determinar que você está conseguindo pingar as outras máquinas da rede e a máquina do instrutor:

```
 # ping -b 192.168.200.255
# ping 192.168.200.254
```

15) Agora que já estamos com a rede configurada vamos tentar acessar a internet. Pingue um site a sua escolha:

```
 # ping www.4linux.com.br
```

Como que o ping sabe qual é o endereço IP do servidor www.4linux.com.br?

16) Configure o arquivo para um cliente DNS:

```
 # vim /etc/resolv.conf
nameserver 200.204.0.10
nameserver 200.204.0.138
```

17) Tente pingar o site novamente e verifique que a resolução de nomes está funcionando

```
 # ping www.4linux.com.br
```

18) Agora que determinamos quem são nossos servidores de nomes podemos configurar também o nosso mini resolvedor de nomes internos:

```
 # vim /etc/hosts
```

Esse arquivo deve sempre conter as informações corretas para o loopback, caso contrário, serviços internos podem deixar de funcionar.

19) Adicione as linhas apropriadas a esse arquivo:

```
127.0.0.1 localhost.localdomain localhost
192.168.200.X microX.treinamento.xxx.br microX
192.168.200.254 gateway.treinamento.xxx.br gateway instrutor
```

20) Realize testes para ver que esses nomes estão funcionando:

```
# ping instrutor
# ping localhost
```

21) Adicione a seguinte linha ao nosso resolvedor de nomes:

```
 192.168.200.254 www.4linux.com.br 4linux
```

22) Pingue o servidor da 4linux:

```
# ping www.4linux.com.br
```

23) Configure dinamicamente o hostname (nome da máquina):

```
# hostname
```

24) Altere o nome da máquina de microX para maqX, de forma que a alteração fique permanente:

```
# vim /etc/hostname
```

Na Red Hat, o hostname fica definido no arquivo /etc/sysconfig/network.

25) Verifique qual é o domínio ao qual a sua máquina pertence:

```
# hostname -d
```

26) Verifique qual é o FQDN (Fully Qualified Domain Name) da sua máquina

```
# hostname -f
```

27) Para que o novo nome seja estabelecido, reinicie a máquina:

```
# shutdown -r now
```

8.12. Exercício Teórico

1) Qual é o papel da máscara na configuração da rede?

2) Qual é o comando que preciso executar para recarregar minhas configurações de rede?

3) Alem do número IP 127.0.0.1, qual é o outro número Ip que pode responder por Localhost?

4) Qual é o papel do Gateway na rede?

5) Aonde podemos configurar o nosso servidor de DNS? Se o mesmo não estiver corretamente configurado o que vai acontecer?

8.13. Laboratório

1) Configure sua placa de rede, de acordo com as especificações do instrutor;
2) Configure também um segundo Ip para essa mesma placa de rede;
3) Tente pingar os colegas nas duas redes configuradas;

Capítulo 9

Manipulando Hardware e Dispositivos

9.1. Objetivos

- Entender o funcionamento do /dev;
- Diferenciar devfs de udev;
- Compreender o processo de montagem de um dispositivo;
- Testar alguns dispositivos presentes em /dev
- Compreender como funciona o particionamento;
- Aplicar um sistema de arquivos a uma partição;

9.2. Dispositivos em Linux

O núcleo do sistema operacional GNU/Linux, o ``kernel'', se comunica com os dispositivos de uma maneira muito interessante: praticamente todos os dispositivos em GNU/Linux são representados por um arquivo correspondente dentro do sistema de arquivos. Exceções a esta regra são as placas de rede.

Alguns assuntos sobre dispositivos foram retirados da apostila pois estão ligeiramente ultrapassados, porém os mesmo podem ser encontrados no Anexo.

O local onde são armazenadas estas representações é o diretório /dev. Uma listagem deste diretório mostrará uma série de arquivos, todos eles representando uma parte do seu computador. A interação com estes arquivos pelo sistema operacional GNU/Linux realiza as leituras dos pedidos, processa estes pedidos e retorna as respostas controlando os sinais enviados aos dispositivos, por exemplo, comandando a placa de vídeo para coordenar as respostas no seu monitor.

9.2.1. Explorando o /dev

Uma diferença marcante entre sistemas Windows e Unix-like é a forma de lidar com partições e dispositivos como unidade de disquete e CD-ROM. Em sistemas Windows desde uma partição no disco rígido a um pen drive o acesso a eles são efetuados utilizando a idéia de "unidades" ou drives, como o drive C: ou A: ou até mesmo uma unidade de rede. Esse tipo de conceito faz com que o usuário final não precise saber o que está por detrás desses equipamentos, simplificando sua utilização ao preço da perda do conhecimento.

Em sistemas como GNU/Linux existe o conceito de dispositivos; praticamente tudo na máquina é tratado como sendo um dispositivo e pode ser acessado pelo seu respectivo arquivo localizado no diretório /dev. Uma exceção a isso é a interface de rede que é tratada diretamente no nível do kernel, não existindo um dispositivo (no /dev) associado a ela.

O diretório /dev consiste de um filesystem especial e pode ser de dois tipos: devfs ou udev. O devfs é o mais antigo tendo sido substituído pelo udev a partir do kernel 2.6.12. Uma das diferenças entre os dois é que no devfs os arquivos de dispositivos são criados uma única vez, dessa forma, o diretório /dev contém os dispositivos para todos os hardwares suportados pelo Linux, não importando se eles estão disponíveis na máquina ou não. Com o udev os dispositivos são criados de acordo com a disponibilidade no sistema. Dessa forma, o diretório contém apenas os arquivos de dispositivo para os hardwares presentes na máquina.

O UDEV não super popula o diretório dev do nosso sistema, alem de nos proporcionar um método de configuração que pode ser encontrado em /etc/udev/

Explorando o diretório /dev você irá se deparar com alguns tipos de arquivos especiais, conhecidos como arquivos de dispositivos. Os tipos existentes são os dispositivos de:

- **bloco -** utilizados para transferência de dados para hardwares de armazenamento de dados como discos rígidos;
- **caracter -** conhecido também como "unbuffered" é utilizado para comunicação com hardwares como mice e terminais;
- **fifo -** conhecido também como pipe nomeado (named pipe) é um dispositivo utilizado para realizar a comunicação entre dois processos independentes;
- **socket -** um dispositivo do tipo socket é utilizado para criar um ponto de comunicação.

Seguindo essa classificação, os dois tipos mais comuns de serem manipulados são os de bloco e de caracter; como exemplos deles temos os devices referentes a dispositivos IDE conectados à máquina (/dev/hda1, por exemplo) e o dispositivo de acesso ao mouse (/dev/psaux, por exemplo).

Outros dispositivos de bloco importantes são os SCSI utilizados não apenas por discos SCSI mas também por dispositivos como USB e SATA, uma vez que são acessados utilizando essa emulação. O nome destes dispositivos são do tipo /dev/sd[letra][número] e seguem a mesma lógica dos dispositivos IDE. Dessa forma, se houver um HD SATA conectado à máquina e mais nenhum outro dispositivo que utilize emulação SCSI, sua localização será o device /dev/sda.

Os nomes dos dispositivos e a maneira como são representados na hierarquia do diretório /dev podem ser bastante difíceis a primeira vista. Com um pouco de prática, a nomenclatura usada fará sentido.

Um mouse USB é representado pelo arquivo /dev/input/mice, que pode ser traduzido como: dispositivo (DEV) de entrada (INPUT) do tipo apontador (MICE outro termo para ``rato'', em inglês). Um mouse PS/2 segue uma nomenclatura um pouco mais complicada e é representada pelo arquivo /dev/psaux, que pode ser

interpretado como dispositivo auxiliar na porta PS.

Para alguns dispositivos como o mouse podemos realmente ver a interação com o arquivo que representa o dispositivo. No exemplo abaixo, usamos o comando cat para mostrar o conteúdo do arquivo de dispositivo de mouse (mexa o mouse depois de pressionar **ENTER** após os comandos abaixo):

Para mice USB:

```
#cat /dev/input/mice
```

Para mice PS/2:

```
#cat /dev/psaux
```

As saídas, ilegíveis para humanos, representam os dados que o sistema operacional GNU/Linux usa para avaliar a movimentação, posicionamento e apertar de botões do mouse.

9.3. Dispositivos de armazenamento
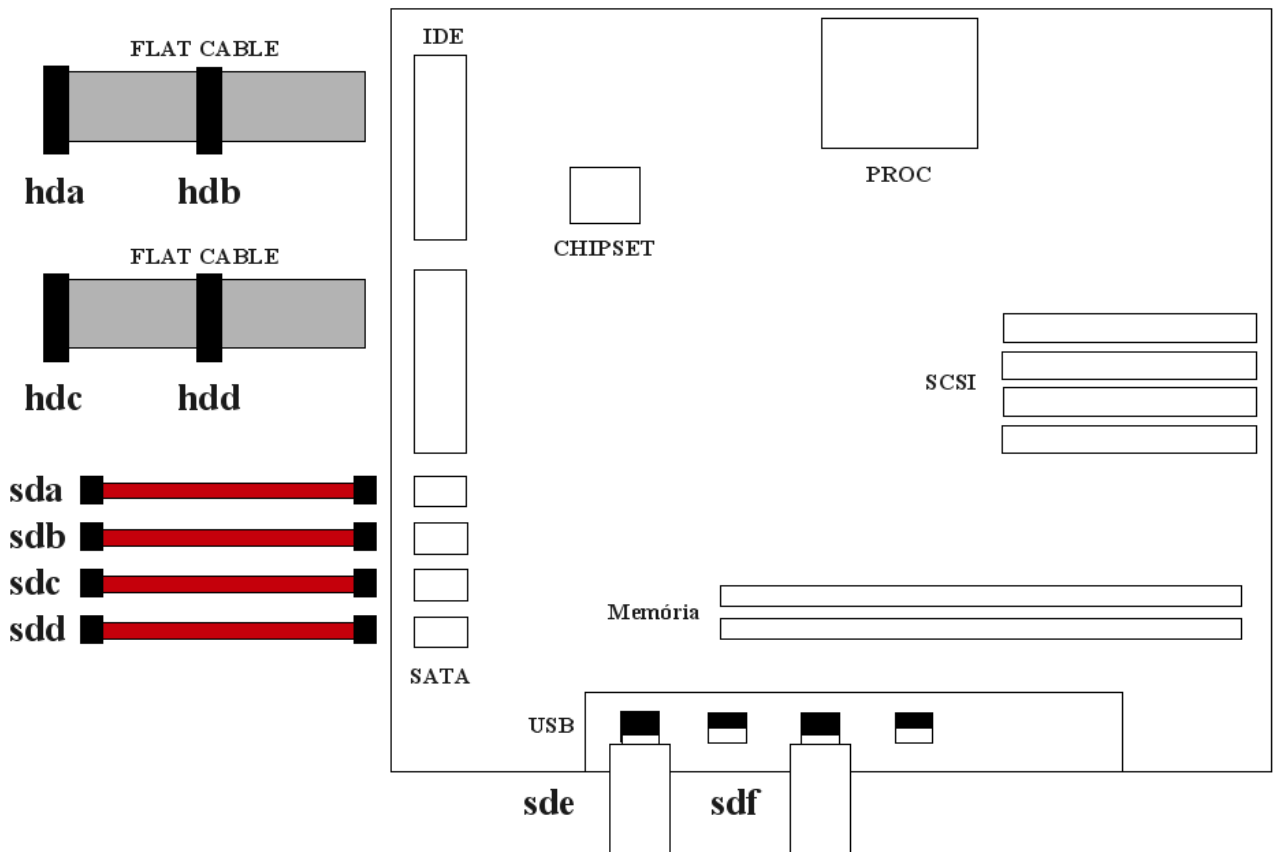
Ilustração 13: Placa mãe

Outro exemplo importante são os dispositivos de armazenamento principais do seu computador, os discos rígidos. Existem três tecnologias principais de discos rígidos, IDE, SATA e SCSI.

Os discos IDE ainda são maioria no mercado, mas a tecnologia vem dando lugar ao padrão SATA. Tanto o padrão IDE como o SATA são considerados econômicos e mais voltados para computadores pessoais ou estações de trabalho.

O discos do padrão SCSI usam uma tecnologia de acesso mais sofisticada, são geralmente mais velozes que similares IDE e SATA e mais robustos. São usados principalmente em servidores e máquinas de alto desempenho.

Os dispositivos IDE são representados na hierarquia do diretório /dev com um padrão que começa com hd. O disco rígido conectado como mestre na controladora principal será designado por hda. Já o escravo, nesta mesma controladora, será representado por hdb. Analogamente, temos hdc e hdd respectivamente para os discos mestre e escravo conectados na controladora secundária.

Por outro lado, o padrão dos dispositivos SATA e SCSI começam por sd. Assim sendo, temos sda para o primeiro dispositivo SATA ou SCSI, sdb para o segundo, etc. Uma controladora SCSI de 8 bits pode comportar até 7 dispositivos, além da própria controladora. Para as de 16 bits, o número máximo de dispositivos é 15.

Podemos verificar o conteúdo de um disco usando novamente o comando cat. Para inspecionar o conteúdo do primeiro disco rígido IDE de um computador, podemos usar o comando abaixo:

```
# cat /dev/hda
```

A saída gerada não parece ter nenhum sentido. Os dados mostrados são aqueles dados gravados no seu disco. Contudo, estão em uma forma que é compreensível apenas pelo sistema operacional.

Uma partição é uma divisão lógica do seu disco rígido, criada por questões de organização, conveniência, flexibilidade ou segurança. Nos sistemas baseados em representação por letras, um disco rígido IDE pode ser dividido, particionado de forma a ser visto com as letras C: e D:. No sistema operacional GNU/Linux, esta mesma divisão levaria aos arquivos representados em /dev/hda1 e /dev/hda2. Ou seja, a primeira partição do disco hda é representada por /dev/hda1 e a segunda é representada por /dev/hda2. Qualquer partição adicional seguiria o mesmo padrão.
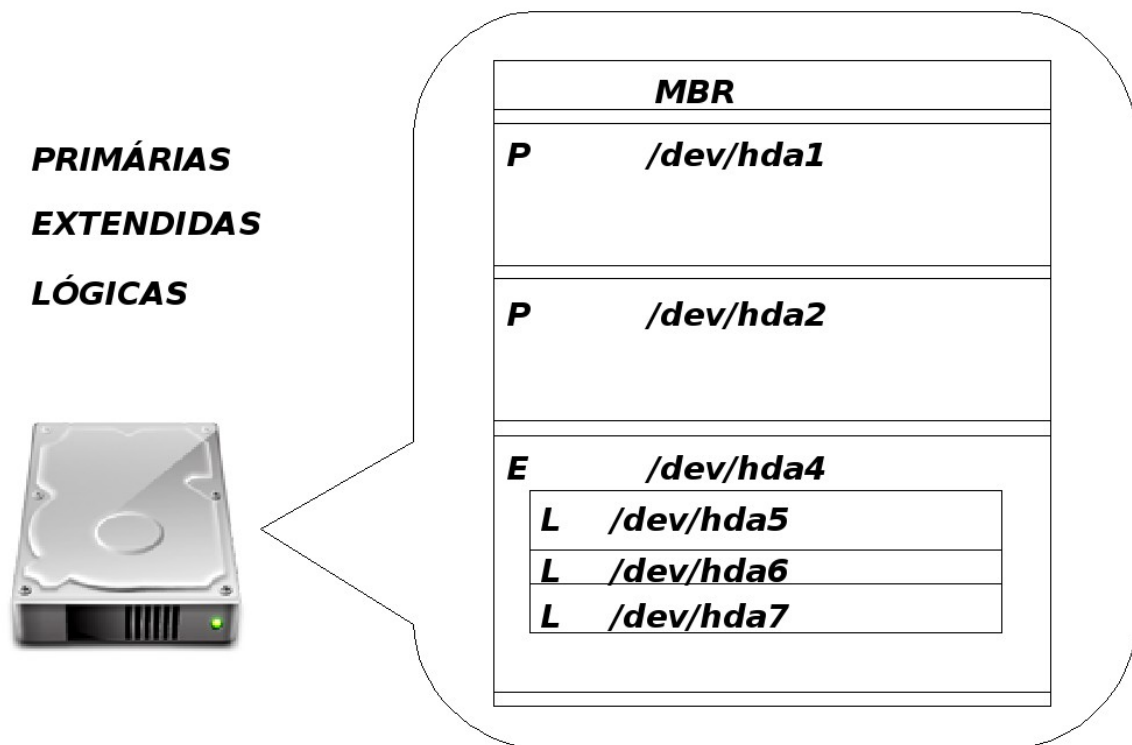
Ilustração 14: Estrutura das Partições

Assim, para inspecionar o conteúdo da primeira partição, pode-se usar o comando abaixo:

```
# cat /dev/hda1
```

Para interromper a saída do comando que pode ser bastante demorada, pressione a combinação de teclas Ctrl C (mantenha a tecla Ctrl pressionada e pressione a letra C). Caso a tela do seu console continue a mostrar caracteres estranhos, digite reset.

O último comando mostra uma saída que seres humanos não conseguem entender. Elas representam a maneira como os dados foram armazenados em /dev/hda1. Para que o sistema operacional GNU/Linux apresente estes dados de uma forma mais legível, é necessário solicitar ao sistema um processo de tradução. Este processo é chamado de montagem de dispositivos.

Então para que a partição /dev/hda1 seja usada, é necessário montar esta partição em algum local e acessá-lo. Este local, que é um diretório no sistema de arquivos, é chamado de ponto de montagem. Podemos montar um dispositivo de armazenamento em qualquer diretório do sistema de arquivos, contudo, existem algumas convenções:

Dispositivos removíveis devem ser montados em /media (em outras épocas em /mnt).

Exemplos:

- Um **cdrom** convencional, representado por /dev/cdrom ou /dev/hdc, pode ser montado em /media/cdrom.
- Um leitor de disquetes, representado por /dev/fd0, pode ser montado em /media/floppy.
- A grande maioria dos dispositivos de bloco USB, são reconhecidos como scsi, e podem localizados em /dev/sda.
- Um Hd Sata também pode ser encontrado em /dev/sda, isso pode variar, depende da porta sata utilizada.

No caso de discos rígidos, uma partição é montada diretamente na raiz do sistema de arquivos ou em um diretório diretamente abaixo da raiz.

9.4. Devices, UUID e Labels

Quando usamos dispositivos seguindo padrões como /dev/hda3 ou /dev/sda5, estamos especificando um dispositivo que pode vir a receber outro nome se houver alguma modificação no disco, isso implica no sistema não mais encontrar a partição especificada pois seu nome foi modificado. Uma alternativa inteligente para isso é utilizar o método UUID - Universally Unique Identifier ou utilizar o método de Lables.

Para descobrirmos o UUID de nossa partições podemos utilizar dois aplicativos: vol_id e blkid

```
 # vol_id -u /dev/sda2
f541a97e-ef19-4e47-b305-b535a75c932a
```

A **flag u** do comando vol_id, nos imprime a UUID referente a uma determinada partição.

```
 # blkid
/dev/sda1: UUID="f541a97e-ef19-4e47-b305-b535a75c932a" TYPE="ext3"
LABEL="MAIN"
/dev/sda3: UUID="7C444A56444A12F6" TYPE="ntfs" LABEL="WIN"
/dev/sda5: TYPE="swap"
/dev/sda6: UUID="69ff8ed5-c09b-49b6-b21d-328e90243efa" TYPE="ext3"
LABEL="HOME"
/dev/sda7: UUID="2c070d34-5c6e-4504-8d4b-9a8fa910548d" TYPE="ext3"
LABEL="STORAGE"
/dev/sda8: UUID="489B-5A22" TYPE="vfat" LABEL="CENTER"
```

Já o comando blkid lista todos os dados relevantes sobre as partições do seu disco.

Há também um outro método de se descobrir essas informações para isso:

```
# ls -l /dev/disk/by-uuid/
lrwxrwxrwx 1 root root 10 2009-03-06 10:41 2c070d34-5c6e-4504-8d4b-
9a8fa910548d -> ../../sda7
lrwxrwxrwx 1 root root 10 2009-03-06 10:41 489B-5A22 -> ../../sda8
lrwxrwxrwx 1 root root 10 2009-03-06 10:41 69ff8ed5-c09b-49b6-b21d-
328e90243efa -> ../../sda6
lrwxrwxrwx 1 root root 10 2009-03-06 10:41 7C444A56444A12F6 ->
../../sda3
lrwxrwxrwx 1 root root 10 2009-03-06 10:41 f541a97e-ef19-4e47-b305-
b535a75c932a -> ../../sda1
```

Mas a resposta gerada não está tão amigável quando as outras. =D

O uso dos métodos de LABEL ou UUID em conjunto com o /etc/fstab é um solução inteligente para o dia-a-dia e para nossa prova.

9.4.1. Usando os dispositivos de armazenamento

Para termos acesso a um arquivo armazenado em mídia removível, é necessário conectar a mídia removível ao seu leitor correspondente e montar o dispositivo adequado.

O comando usado para montar dispositivos é o mount. Sem o uso de nenhum parâmetro, ele mostra os dispositivos de armazenamento que estão montados em seu computador junto com a configuração usada para montá-los.

```
# mount
```

Existem muitos comandos para descobrirmos o que temos conectados em nossas máquinas, dentre eles: lspci, lsusb e lsscsi

Para montar um dispositivo de armazenamento em seu ponto de montagem, o comando mount pode ser usado da seguinte forma:

```
# mount -t <tipo> -o <opções> <dispositivo> <ponto-de-montagem>
```

Para que seja possível acessar o conteúdo de algum dispositivo precisamos de quatro itens básicos:

- saber qual o nome do dispositivo que será acessado;
- saber qual o filesystem que ele está utilizando;
- ter um ponto de montagem;
- ter permissão de montagem;

O método mais garantido de encontrar o nome de um dispositivo é realizar uma busca na saída do comando dmesg; por exemplo, se desejarmos determinar qual o nome do device do CD-ROM, podemos tentar:

```
# dmesg |grep ATAPI
```

As informações providas pelo comando dmesg são providas pelo arquivo /var/log/messages

Uma vez determinado o nome do dispositivo podemos, realizar outra procura no dmesg, mas agora com o nome do dispositivo, e determinar qual filesystem está utilizando.

Se não existir um ponto de montagem, basta criar um diretório vazio no local apropriado; em geral no /media ou /mnt e executar o comando para montá-lo. Por exemplo:

Para um cdrom, a sintaxe do comando seria:

```
# mount -t iso9660 /dev/cdrom /media/cdrom
```

Na maioria dos ambientes gráficos, este processo de montagem é automatizado. A simples inserção ou conexão de mídias removíveis faz com que elas sejam montadas e acessíveis pelos navegadores de arquivos gráficos.

Para desmontar um dispositivo, o comando usado é o umount. Neste caso é possível usar como parâmetro o ponto de montagem ou o dispositivo:

Por exemplo:

```
# umount /media/cdrom
```

ou de forma equivalente:

```
# umount /dev/cdrom
```

Uma alternativa para a montagem aleatória do sistema, é utilizar o pacote autofs que nos prove toda uma estrutura configurável para os dispositivos. Para usar autofs seu sistema precisa usar Kernel 2.6 e ter a partição /dev com udev

9.5. Criando Partições no HD

Agora que já sabemos como montar um dispositivo precisamos saber como criar uma partição manualmente. Para isso, há duas ferramentas importantes disponíveis em sistemas GNU/Linux, são elas fdisk e cfdisk.

Conhecer esses particionadores é muito importante, anote mais uma ai: Gparted

9.5.1. Particionamento com FDISK

O particionador fdisk é o mais completo dos particionadores e que em geral resolve nossos problemas quando eles ocorrem.

Fazendo a chamada a esse programa podemos ver a seguinte tela inicial:

```
 # fdisk /dev/hda
The number of cylinders for this disk is set to 14593.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
(e.g., DOS FDISK, OS/2 FDISK)

Command (m for help):
```

Pressionando a tecla **m** para obtermos um help, veremos a seguinte saída:

```
 Command (m for help): m
Command action
a   toggle a bootable flag
b   edit bsd disklabel
c   toggle the dos compatibility flag
d   delete a partition
l   list known partition types
m   print this menu
n   add a new partition
o   reate a new empty DOS partition table
p   print the partition table
q   quit without saving changes
s   create a new empty Sun disklabel
t   change a partition's system id
u   change display/entry units
v   verify the partition table
w   write table to disk and exit
x   extra functionality (experts only)

Command (m for help):
```

Para criarmos uma nova partição devemos antes ver se temos espaço disponível para isso, ou seja, precisamos imprimir a tabela de partições utilizando a letra p. Se houver espaço disponível para a criação de uma nova partição basta pressionar a letra n e informar o tipo da partição (primária ou estendida) e seu tamanho.

9.5.2. Particionamento com CFDISK

A ferramenta cfdisk não é tão completa quanto o fdisk mas é um pouco mais "user friendly". Para acessá-la basta executar o comando:

```
 # cfdisk /dev/hda
```

Uma vez executado esse comando, a tela do cfdisk se abrirá como mostrado na figura:

```
                              cfdisk 2.11n

                          Disk Drive: /dev/hda
                        Size: 10205282304 bytes
          Heads: 255   Sectors per Track: 63   Cylinders: 1240

   Name        Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
   hda1        Boot        Primary   Linux ext3                          16.46
   hda2                    Primary   Linux ext3                         501.75
   hda3                    Primary   Linux swap                         254.99
   hda5                    Logical   Linux ext3                        4499.23
   hda6                    Logical   Linux ext3                         296.12
   hda7                    Logical   Linux ext3                         197.41
   hda8                    Logical   Linux ext3                        3997.49
                           Logical   Free Space                         435.94



     [Bootable]  [ Delete ]  [  Help  ]  [Maximize]  [ Print  ]
     [  Quit  ]  [  Type  ]  [ Units  ]  [ Write  ]

                 Toggle bootable flag of the current partition
```

Ilustração 15: cfdisk

A utilização do cfdisk é bastante intuitiva, utilizando as setas para cima e para baixo você navega pela listagem das partições e, utilizando as setas para a esquerda e direita, você navega pelo menu na parte inferior da tela.

Para criar uma nova partição basta selecionar na listagem de partições a linha que contém espaço livre e entrar na opção **New** no menu inferior. Se ainda for possível criar partições primárias surgirá a pergunta pelo tipo da partição, caso contrário, surgirá a pergunta para especificar quanto espaço deve ser destinado para essa partição.

Após realizar todas as alterações, escolha, no menu inferior, a opção Write. Uma pergunta pedindo que você confirme as alterações irá aparecer. Sua resposta deve ser ``sim'' ou ``não'' com todas as três letras!! Afinal, você é o root e sabe o que está fazendo! :)

Pronto, criadas as partições precisamos aplicar um filesystem.

9.6. Aplicando um Filesystem

Para que possamos gravar informações de forma estruturada na partição que acabamos de criar precisamos aplicar um filesystem a ela. Sim, aplicar um filesystem, NÃO formatá-la!!!

Formatar é o processo de preparar a mídia magnética, como discos rígidos e disquetes, para receber informação. Esse tipo de preparo é de baixo nível e consiste em ``desenhar'' as trilhas e setores na mídia em questão. Aplicar o filesystem significa criar uma estrutura lógica acima dessas trilhas e setores que permita organizar seus arquivos em uma estrutura de diretórios e subdiretórios.

Vamos conhecer alguns tipos de FileSystem

- **ext2 -** Um dos primeiros filesystens do linux;
- **ext3 - Evolução** do ext2, mas com a técnica de Journal
- **reiserfs -** Ótimo sistema de arquivos para arquivos menores que 4MB
- **xfs -** Usado geralmente em banco de dados, tem suas vantagens com objetos muito grandes.

As ferramentas de manutenção do xfs conhecidas como xfs-tools, podem de dar alguns pontos positivos na prova. O pacote referente no Debian se chama xfsprogs

Para criarmos um filesystem em uma partição devemos escolher o tipo de filesystem e utilizar o comando mkfs cuja forma de utilização básica é a seguinte:

```
mkfs -t tipo_do_FS <dispositivo>
```

Leitura sugerida: man mkfs

Sendo que o filesystem que você pode escolher para criar no device deve ser

suportado pelo kernel e deve ter seu software instalado. Para consultar quais filesystem estão com suporte no kernel basta consultar o arquivo /proc/filesystems.

Dessa forma, podemos exemplificar a criação de um filesystem em um dispositivo utilizando o seguinte comando:

```
# mkfs -t ext3 /dev/hdb1
```

Aplicado o filesystem, só falta criar o ponto de montagem e montar!

Ao contrário do que pensam os file systens não mordem, e podem ser grandes aliados na prova, principalmente no termo migração de file system.

9.7. Arquivos de Informações de Filesystems

Na seção sec:mount você aprendeu a montar um dispositivo de forma completa e manual, entretanto, há um arquivo que facilita a nossa vida, o /etc/fstab. Nele devem estar as informações a respeito da montagem de todos os filesystems do sistema, veja um exemplo a seguir:

```
<file system> <mount point><type> <options> <dump> <pass>
proc      /proc proc defaults     0         0
/dev/hda1      /boot    ext3 defaults  0         1
/dev/hda2      /        ext3 defaults,errors=remount-ro 0 2
/dev/hda3     none  swap     sw        0          2
/dev/hda5 /usrext3defaults   0          2
/dev/hda6     /var      ext3 defaults  0 2
/dev/hda7     /tmp      ext3 defaults  0          2
/dev/hda8     /home     ext3 defaults  0          0
UUID=be35a709-c787-4198-a903-d5fdc80ab2f8 /media/chas ext3
relatime,errors=remount-ro 0 1
```

As informações que devem ir nesse arquivo, de acordo com o número da coluna são:

- **localização do filesyste**m, em geral o device ou endereço de rede;
- **ponto de montagem**;
- **tipo do filesystem**, ext3, reiserfs, xfs, etc;
- **opções de montagem** (defaults = rw, suid, dev, exec, auto, nouser e async). Ver man mount;
- aceita os valores 0 ou 1 e informa que, havendo um sistema de backup **(dump)** configurado, deverá ser feito o seu backup;
- aceita os valores de 0 a 2 e informa que deverá ser realizada a checagem **(pass)** de integridade do sistema de arquivos. O valor zero desativa a funcionalidade, o valor 1 deve ser especificado apenas para o / e o valor 2 deve ser especificado para quaisquer outros sistemas de arquivos.

Sendo assim, o fstab armazena as informações dos dispositivos comumente acessados, como as partições do sistema, discos removíveis e alguns dispositivos USB, entretanto, não mostra informação alguma a respeito de quais dispositivos estão montados neste exato momento.

Essa informação pode ser obtida acessando o arquivo /etc/mtab ou /proc/mounts; ambos os arquivos são uma tabela atualizada em tempo real e que mostra quais dispositivos estão montados e com quais

parâmetros.

9.8. Configurações de Teclado e Mouse no Console

Imagine que você instalou uma máquina na sua casa e o seu teclado é um teclado padrão brasileiro -- ABNT2 -- e você chegou ao cliente e ele só possui teclados com layout americano; como resolver o problema?

É possível utilizar o comando loadkeys para alterar o layout de teclado durante a sessão mas, essa alteração será temporária. Para trocar definitivamente o padrão de layout de teclado da máquina, o arquivo em /etc/console/boottime.kmap.gz deve ser alterado utilizando o comando kbd-config.

Além da configuração apropriada de layout de teclado, pode ser interessante configurar o mouse em modo texto a fim de facilitar o trabalho. O programa chamado gpm é o que dá o suporte ao mouse em terminais texto.

9.9. Prática Dirigida 1

1) Altere o layout de teclado para utilizar o padrão americano:

```
# loadkeys -d us
```

Tente utilizar a tecla ``ç''

2) Volte o layout de teclado para o padrão br-abnt2:

```
# loadkeys -d br-abnt2
```

3) Altere o layout de teclado padrão do sistema. Você tem duas opções:

```
 # kbd-config
# dpkg-reconfigure console-data
```

9.10. Para aprofundar o assunto

Parâmetros de montagem:

```
 # info mount
```

O arquivo /etc/fstab:

```
 # info fstab
```

9.11. Prática Dirigida 2

1) Determine qual o dispositivo associado ao "hd", tanto o CD-ROM, quanto o Disco Rígido:

```
 # dmesg |grep hd
```

2) Coloque um CD no drive e torne o conteúdo acessível no diretório /media/cdrom:

```
 # mount -t iso9660 <dispositivo> /media/cdrom
```

3) Verifique que o dispositivo foi montado:

```
 # mount
# df -h
# cat /etc/mtab
# cat /proc/mounts
```

4) Entre no diretório e explore o conteúdo do CD:

```
# cd /media/cdrom ; ls
```

5) Desmonte o CD:

```
# umount /media/cdrom
```

Deu erro?? Por quê??

6) Saia do diretório /media/cdrom:

```
# cd
```

7) Tente desmontá-lo novamente:

```
# umount /media/cdrom
```

Agora sim!

8) Monte o CD novamente, entre no diretório do ponto de montagem e, de dentro dele, abra uma nova shell:

```
# mount -t iso9660 <dispositivo> /media/cdrom
# cd /media/cdrom
# bash
```

9) Saia do diretório e desmonte o CD:

```
# cd
# umount /media/cdrom
```

Funcionou?? E agora??

Agora que já sabemos montar dispositivos, vamos aprender a cria partições.

10) Utilizando o particionador 'cfdisk' crie uma nova partição de 'swap' e uma nova partição linux:

As novas partições devem ter os seguintes tamanhos:

- **/dev/hda9 -** partição tipo swap com 256MB;

- **/dev/hda10 -** nova partição com 1000;
- espaço vazio não particione.

```
# cfdisk /dev/hda
```

Após criar as novas partições será necessário rebotar a máquina para que a nova tabela de partições seja relida. No nosso caso, a versão do Kernel é a 2.6.18, então estamos utilizando udev, mesmo sendo udev, temos que reiniciar.

Com a tabela de partições atualizadas podemos aplicar os filesystems e paginar a nova partição de swap.

11) Prepare a partição de swap:

```
# mkswap <dispositivo>
```

12) Ative essa nova partição de swap:

```
# swapon <dispositivo>
```

O filesystem que desejamos aplicar a uma das novas partições é o etx3. Para que possamos realizar essa tarefa devemos determinar se o nosso kernel suporta este filesystem e se o software necessário está instalado.

13) Determine se o ext3 pode ser utilizado, ou seja, tem suporte no kernel:

```
# ls /proc/filesystems
```

14) Instalados os softwares podemos aplicar o ext3 à nova partição:

```
# mkfs -t ext3 <dispositivo>
```

Uma vez que o filesystem foi aplicado à partição, vamos torná-lo acessível por

meio do diretório /backup.

15) Crie o ponto de montagem /backup:

```
# mkdir /backup
```

16) Teste a montagem do novo filesystem:

```
# mount -t ext3 <dispositivo> /backup
```

17) Verifique se a partição foi montada e se o swap está em uso:

```
# mount
# df -h
# cat /etc/mtab
# cat /proc/mounts
# cat /proc/swaps
```

17) Crie um arquivo dentro do /backup:

```
# touch /backup/README
# echo "Partição de Backup" >> /backup/README
```

18) Coloque as entradas no fstab para que o novo swap e a partição de backup sejam montadas na hora do boot:

```
# vi /etc/fstab
<dispositivo> none swapsw 0 0
<dispositivo> /backup ext3 defaults 0 0
```

O arquivo /etc/fstab é criado automaticamente quando o Linux é instalado. Abaixo algumas das suas opções, importantes:

9.12. Exercícios Teóricos

1) Qual o nome do arquivo de dispositivo que é a oitava partição do HD conectado como slave na segunda controladora IDE? Forneça o caminho completo para ele.

2) Qual a diferença entre os arquivos /etc/fstab e /etc/mtab?

3) Qual comando pode ser utilizado para determinar se o mouse está conectado ao /dev/psaux? Por quê?

4) Quando o comando mount pode ser executado com sucesso especificando apenas o nome do dispositivo ou apenas o ponto de montagem?

5) Qual deve ser a linha no fstab para que não seja possível executar um programa ou script a partir da partição /home?

6) Quais parâmetros dos comandos fdisk e cfdisk podem ser utilizados para imprimir na tela a tabela de partições sem entrar no programa propriamente dito? Dê os comandos completos como resposta.

9.13. Laboratório

1) Crie um arquivo imagem do CD-ROM;
2) Monte este arquivo imagem;
3) Acesse este sistema de arquivos que você montou, e veja o seu conteúdo.

Capítulo 10

Administração de Usuários

10.1. Objetivos

- Aprender a gerenciar usuários e grupos;
- Conhecer o sistema de permissões;
- Conhecer o funcionamento do umask;
- Entender as permissões especiais;

O GNU/Linux é um sistema multiusuário e portanto, possui um esquema de permissões que provê a privacidade e/ou compartilhamento de arquivos entre usuários. Na verdade, esse esquema de permissões é parte fundamental do sistema. Neste capítulo, iremos aprender sobre ele e também como criar e remover contas de usuários.

10.2. Gerenciamento de usuários

Quando começamos a trabalhar com usuários no sistema GNU/Linux podemos dividi-los em 3 categorias:

- **Usuário Administrador (Super Usuário):** usuário conhecido como root no sistema. É esse usuário que controla todo o sistema e não possui nenhuma restrição. Mas devemos ter uma certa cautela ao usá-lo pois com qualquer deslize podemos danificar todo o sistema.
- **Usuários de Sistema:** são aqueles que não precisam logar no sistema, são utilizados para controlar serviços. Esse usuários não devem possuir senhas nem Shell's válidas. Um exemplo desses usuários é o usuário www-data que é usado exclusivamente para controlar o servidor web Apache.
- **Usuários comuns:** são utilizados para trabalhar no sistema

GNU/Linux. São contas criadas para aqueles que utilizam ou operam o sistema. É sempre aconselhável que cada usuário comum ou administrador tenha sua própria conta e só utilize a conta root para administração do sistema.

Tanto para o usuário root quanto para o usuário comum, é sempre aconselhável criar uma boa política de criação de senhas para que um possível invasor não se aproveite de um usuário com uma senha fraca, mesmo que seja um usuário comum, pois isso seria o primeiro passo para o invasor escalar privilégios no sistema e virar o usuário administrador root. Evite usar senhas com datas de aniversário, casamento e outras datas que são fáceis de serem descobertas e evite usar palavras do dicionário. Uma boa dica é mesclar a senhas com letras maiúsculas e minúsculas, caixa alta e caixa baixa e caracteres especiais.

Alguns sistemas GNU/Linux podem ter usuários que chamamos de administradores. Esses usuários não vêm configurados por padrão, eles são usuários normais mas que possuem alguns privilégios a mais em algumas aplicações.

Para que os usuários comuns e root tenham acesso ao sistema e consigam trabalhar normalmente, são necessários 5 elementos.

- Nome;
- Senha;
- Diretório Home;
- Shell;
- Grupo Primário;

Devemos ter em mente que um usuário sempre deve estar vinculado a um grupo, pois isso afeta diretamente a questão de permissões dentro do sistema.

10.3. Permissões

Cada arquivo no sistema possui três permissões básicas:

- **r (4) -** read – para leitura;
- **w (2) -** write para escrita;
- **x (1) -** execute para execução;

A cada permissão é atribuído um valor, mostrado entre parênteses, que ser á utilizado para atribuição de permissões.

Além disso, cada arquivo contém três conjuntos de permissões, sendo elas:

- **permissão do dono (u) -** user do arquivo;
- **do grupo (g) -** group ao qual o arquivo pertence;
- **outros (o) -** others; aqueles que não pertencem ao grupo e não são os donos do arquivo;

Sendo assim, considere a seguinte saída do comando ls -l:

```
-rw-r--r-- 1 root root 0 Jan 15 09:52 arquivo
```

Para um arquivo e para um diretório:

```
drwxr-xr-x 2 root root 4096 Jan 15 09:52 diretório
```

Vamos entender o que essas linhas significam, o primeiro caractere pode ser:

- **- -** indicando que o que está sendo listado é um arquivo comum;
- **d -** indicando um diretório;
- **l -** indicando um link simbólico;
- **p -** indicando um pipe nomeado;
- **s -** indicando um socket;
- **c -** indicando um dispositivo de caractere;
- **b -** indicando um dispositivo de bloco.

Os próximos três conjuntos de três caracteres indicam as permissões do usuário dono do arquivo, permissões de grupo e permissões para outros usuários. Nesses três conjuntos, se o caractere encontrado for um `-' (hífen) significa que a permissão está ausente, ou seja, não há a respectiva permissão. Se alguma ou todas as letras (r, w e x) forem encontradas, indicará as permissões que o arquivo tem.

Seguindo o conjunto de permissões, há um número que indica a quantidade de links simbólicos que o arquivo ou diretório tem. Após o número de links, vem a indicação do usuário dono do arquivo seguido do grupo ao qual ele pertence.

A atribuição de permissões é realizada utilizando o comando chmod. Há duas sintaxes possíveis. A primeira delas é a literal. Vejamos o exemplo abaixo:

```
# chmod u+rw arquivo
```

O parâmetro u+rw é que define o esquema de permissões. A primeira letra indica a qual(is) usuário(s) as permissões estão sendo alteradas. Usamos a letra u para indicar o próprio dono, g para indicar o grupo, o para outros e ainda a letra a para indicar todos.

O caractere seguinte poderá ser um sinal + para garantir a permissão ou - para retirar a permissão. Por fim, detalhamos a permissão: A letra r significa leitura,

w escrita e x execução, como era de se esperar.

Assim, o exemplo anterior garante as permissões de leitura e escrita para o usuário dono do arquivo. Vejamos mais um exemplo:

```
# chmod g-w arquivo
```

Este comando retira a permissão de escrita para os usuários pertencentes ao mesmo grupo ao qual o arquivo pertence. As demais permissões não são alteradas.

A segunda sintaxe é a forma numérica. Neste caso, o parâmetro que define as permissões é composto de três números de 0 a 7 que correspondem às permissões para o usuário dono, para o grupo e para outros. Cada número é formado pela soma das permissões atribuídas, sendo que execução vale 1, escrita vale 2 e leitura 4.

A tabela abaixo resume esse esquema:

r (4)	w (2)	x (1)	Total	Permissões
0	0	0	**0**	**- - -**
0	0	1	**1**	**- - x**
0	1	0	**2**	**- w -**
0	1	1	**3**	**- w x**
1	0	0	**4**	**r - -**
1	0	1	**5**	**r - x**
1	1	0	**6**	**r w -**
1	1	1	**7**	**r w x**

Vejamos um exemplo:

```
# chmod 640 arquivo
```

Neste caso, estamos atribuindo permissão de leitura e escrita 6 (r=4 + w=2) ao usuário dono, leitura 4 (r=4) ao grupo e 0 (sem permissões) a outros usuários.

É importante observar que quando usamos a forma literal, alteramos apenas o parâmetro especificado, não alterando as demais permissões. Já na forma numérica, alteramos todas as permissões simultaneamente.

10.3.1. Exemplos de permissões

Comando para atribuir permissão total a um arquivo chamado chmod

```
chmod 777 dontdothis.never
```

ou

```
chmod a+rwx dontdothis.never
```

ou

```
chmod u+rwx,g+rwx,o+rwx dontdothis.never
```

Não fazer isso em, nenhum tipo de arquivo, isso é apenas um exemplo!!!

Comando para retirar a permissão de escrita de todos os usuários para o arquivo noexecute.never

```
chmod a-w noexecute.never
```

Comando para alterar a permissão padrão do arquivo Dori.jad para que todos os usuários apenas possam ler.

```
chmod 444 Dori.jad
```

10.4. Registro de usuários no sistema

Há quatro arquivos básicos que dizem respeito à administração de usuários, sendo eles:

- **passwd -** contém as informações dos usuários;
- **shadow -** contém as informações das senhas dos usuários;
- **group -** contém informação dos grupos e usuários que fazem parte dele;
- **gshadow -** contém informações a respeito das senhas de grupo.

Leitura Sugerida:

```
# passwd man 5 passwd
# shadow man 5 shadow
# group man 5 group
# gshadow man 5 gshadow
```

10.4.1. Arquivo /etc/passwd

Cada usuário cadastrado no sistema é identificado por uma linha do arquivo /etc/passwd. Os campos são separados pelo caractere ``:'' (dois pontos). O formato do arquivo /etc/passwd é o seguinte:

```
usuário:x:1000:1000:Usuário da Silva,8111-1234:/home/usuário:/bin/bash
```

Onde:

- **Campo 1 -** Login do usuário;
- **Campo 2** - Referência da senha do usuário, pois ela fica armazenada em outro arquivo.
- **Campo 3 -** UID: O UID (User Identify) é o número de identificação do usuário. Essa identificação é dividida conforme a categoria dos usuários:

- **UID 0 -** É o número do usuário administrador root.
- **UID de 1 a 999 -** São os números para usuários de sistema.
- **UID de 1000 a 65535 -** São os números para usuários normais.

Essas definições de usuários de sistema e usuários normais podem variar nas distribuições, somente o UID 0 é padrão em todas as distribuições.

- **Campo 4 -** GID: O GID (Group Identity) é o número de identificação do grupo primário do usuário. Essa identificação é também dividida em 3 categorias como o UID:
  - **GID 0 -** É o número do grupo administrador root.
  - **GID de 1 a 999 -** São os números para grupos de sistema.
  - **GID de 1000 a 65535 -** São os números para grupos normais.
- **Campo 5 -** Comentários e informações adicionais sobre o usuário;
- **Campo 6 -** Diretório pessoal;
- **Campo 7 -** Shell;

Usar o comando getent para visualizar esses arquivos muito importante.e

10.4.2. Arquivo /etc/shadow

As senhas dos usuários ficam armazenadas no arquivo /etc/shadow conhecido como senhas sombras (shadow passwords). As senhas ficam nele pois é um arquivo mais seguro que o arquivo /etc/passwd. No arquivo /etc/passwd qualquer usuário pode visualizá-las e copiá-las para outro diretório ou máquina remota. No caso de /etc/shadow, suas permissões são muito mais restritas não permitindo que sejam copiadas e nem visualizadas. Isso é uma grande ajuda na questão de segurança pois se as senhas estivessem no próprio /etc/passwd seria muito fácil para um invasor com usuário comum copiar esse arquivo para outro servidor e aplicar uma ferramenta de brute force para quebrar as senhas.

O suporte a senhas shadow costuma vir por padrão nas distribuições. Em algumas delas, se forem instaladas no modo expert, pode-se optar por ativar ou não

esse suporte. É sempre recomendado deixar as senhas shadow ativadas.

Caso encontremos algum servidor GNU/Linux sem as senhas shadow configuradas, podemos utilizar o comando pwconv para ativá-las e pwunconv para desativá-las.

A questão das senhas shadow e os comandos ***pwconv e pwunconv*** *são cobrados com frequência nas provas da LPI.*

O arquivo shadow não trata somente a questão de segurança de senhas. Ele também trata de políticas de contas do usuário, como, por exemplo, por quantos dias a conta de um usuário é válida, quando vai expirar, quando deve ser a troca de senha e alguns outros parâmetros que podem ser alterados na mão ou usando o comando chage.

10.5. Levantamento de informações dos usuários

10.5.1. Chage

O comando chage configura algumas informações como: data de validade do password, data de aviso de troca de senha dentre outras.

```
# chage -E 03/08/2009 flavio
# chage -l flavio
Last password change                                    : Feb 27, 2009
Password expires                                : never
Password inactive                               : never
Account expires                                         : Mar 08, 2009*
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
```

10.5.2. Comando id

O comando id mostra as informações de UID, GID e grupos secundários dos usuários. Para ver essas informações do usuário corrente, fazemos da seguinte forma:

```
 # id
```

Para ver as informações de qualquer outro usuário usamos a seguinte

sintaxe:

```
# id [usuário]
```

10.5.3. Comando finger

O comando finger é mais amigável e nos traz maiores informações como: Login, Nome, Diretório home, Shell e os horários dos últimos logins que esse usuário realizou.

A sintaxe do comando finger é a seguinte:

```
 # finger [usuário]
```

10.5.4. Comando users

O comando users mostra de maneira bem simples os usuários que estão logados no sistema. A sintaxe do comando users é a seguinte:

```
# users
```

10.5.5. Comando who

O comando who mostra quais usuários estão logados na máquina. Traz informações adicionais sobre qual terminal está sendo utilizado, o momento e a partir de qual máquina foi feito o login de cada usuário.

```
# who
```

10.6. Comando w

O comando w é similar ao who, mas traz também informações sobre o que cada usuário está fazendo, tanto local quanto remotamente. Esse comando é muito útil para ver se não existem conexões indevidas em nosso sistema.

A sintaxe do comando w para visualizar todos os usuários logados é a seguinte:

```
# w
```

Para visualizar se somente um usuário está logado a sintaxe é a seguinte:

```
# w [usuário]
```

10.7. Criando Usuários

10.7.1. Comando adduser

O comando adduser é script customizado que trabalha com o comando useradd. O adduser é bastante utilizado por administradores que precisam cadastrar usuários no formato tradicional, ou seja, com nome, senha e grupo definido.

No caso do Red Hat, o comando adduser possui uma sintaxe mais complexa, equivalente ao useradd do Debian.

Este comando pode ser usado de várias formas, mas a sintaxe mais comum de se trabalhar é a seguinte:

```
# adduser [usuário]
```

Dessa maneira ele adicionará o usuário, já pedindo para definir sua senha e as informações adicionais. Automaticamente, ele já cria um grupo com o mesmo nome do usuário e copia todos os arquivos que estão no diretório /etc/skel para o diretório home do usuário.

Podemos também adicionar usuários através do comando useradd, que é um pouco mais complexo e precisa de alguns parâmetros a mais:

```
# useradd teste
```

Para complementar a seção criando usuários é muito interessante olhar o arquivo **/etc/adduser.conf.**

10.8. Adicionar um usuário a outro grupo

Um usuário sempre deve pertencer a um grupo primário, mas pode ser adicionado a grupos secundários, normalmente usado dentro de uma estrutura empresarial onde os usuários precisam pertencem a vários grupos para terem acessos a arquivos de outros setores.

10.8.1. Comando gpasswd

O comando gpasswd pode ser utilizado para definir a senha de um grupo, e com alguns parâmetros pode adicionar ou remover um usuário de um grupo secundário. Utilizando a opção -a podemos adicionar um usuário a um grupo secundário e a opção -d para remover um usuário de um grupo secundário.

Para adicionar um usuário em um grupo secundário a sintaxe é a seguinte:

```
# gpasswd -a [usuário] [grupo]
```

Para remover um usuário de um grupo secundário a sintaxe é a seguinte:

```
# gpasswd -d [usuário] [grupo]
```

10.9. Modificando usuários

A modificação de usuários é limitada ao usuário root. Iremos aprender aqui como mudamos alguns parâmetros que são necessários no dia-a-dia, como troca de senhas, grupos e controle de login.

10.9.1. Comando passwd

Depois do usuário criado podemos usar alguns comandos para modificar a conta dele. O primeiro será o passwd. O passwd possibilita adicionar ou modificar a senha de um usuário. As sintaxes que podem ser utilizadas nesse comando são as seguintes:

Para modificar a senha do usuário corrente:

```
# passwd
```

Observação: Caso esteja modificando a senha de um usuário normal, primeiro será solicitada a senha corrente para depois digitar a nova senha. Isso não acontece com o usuário root, que pode definir a nova senha diretamente, tanto para ele quanto para os outros usuários.

Para modificar a senha de outro usuário:

```
# passwd [usuário]
```

10.9.2. Comando usermod

Outro comando que pode ser utilizado para modificar parâmetros do usuário é o usermod. Ele possibilita modificar qualquer tipo de informação relativa ao usuário. Um dos parâmetros que pode ser modificado é o grupo primário, usando-se a opção -g. Com a opção -G, podemos alterar os grupos secundários.

A sintaxe para modificar o grupo primário de um usuário é a seguinte:

```
# usermod -g [grupo] [usuário]
```

As alterações podem ser visualizadas no arquivo /etc/passwd no campo GID.

Para mudarmos o campo de descrições dentro do arquivo passwd, precisamo usar o comando usermod com a flag c.

```
 # usermod -c teste kiko
# getent passwd | grep kiko
kiko:x:1000:1000:teste:/home/kiko:/bin/bash
```

Saber saber as diferenças entre as flags l e L do usermod pode te ajudar na prova. Então aqui vai: A flag l modifica o login de um determinado usuário enquanto a flag L bloqueia a conta acrescentando um ! no inicio da linha do usuário no passwd.

10.10. Alteração do Dono e Grupo

Como já vimos, cada arquivo e diretório possui um dono e um grupo. Para podermos alterá-los podemos utilizar os comandos chwon e chgrp como nos exemplos a seguir.

```
 # chown euvaldo planta
```

Esse comando irá alterar o dono atual do arquivo 'planta' para o usuário euvaldo. Para alterar o grupo basta o seguinte comando:

```
 # chgrp cacho planta
```

Esse comando irá alterar o grupo atual do arquivo planta para o grupo 'cacho'.

10.11. Removendo usuários

A remoção de usuários pode ser feita de duas formas. A primeira é mantendo o diretório home do usuário e a segunda, apagando o diretório home. É aconselhável

que se remova o diretório do usuário para que um próximo usuário adicionado ao sistema não acabe como dono daquele diretório, já que a delegação de UID's é seqüencial. Mas para remover o usuário com o seu diretório, também é aconselhável fazer um backup de tudo o que aquele usuário possuía ou transferir todos os arquivos para o responsável.

A sintaxe para remover o usuário e manter o seu diretório home é a seguinte:

```
# userdel [usuário]
```

Para remover o usuário e o seu diretório home, é necessário utilizar a opção -r na seguinte sintaxe:

```
# userdel -r [usuário]
```

10.12. Umask

O umask altera o valor da máscara de criação de arquivos e diretórios. Para calcular a umask para um diretório, pegue a permissão total que um diretório pode chegar, 777. Subtraia 022 de 777 (valor padrão de umask do sistema).

Para calcular a umask para um arquivo, pegue a permissão total que um objeto pode chegar, que é 777. Subtraia 111 de 777 (valores de execução do arquivo) e de 022 (valor padrão de umask do sistema). O valor padrão da umask fica armazenada no arquivo /etc/profile.

Lembre-se da regra de calculo de umask, pensar da forma que o sistema funciona pode te confundir na prova:
Para diretórios: Sempre substituir de 777;
Para arquivos: Verificar umask, se o numero for impar, subtrair de 7, senão subtrair de 6.

10.13. Permissões Especiais

Há um conjunto especial de permissões, conhecido também como bits especiais, sendo eles:

Nome	Significado	Valor
SUID	Set User Id Bit	4
SGID	Set Group ID Bit	2
Sticky Bit	Sticky Bit	1

O SUID bit é atribuído a um executável quando desejamos que um usuário qualquer execute o comando (com SUID bit ligado) com as permissões do usuário dono do comando. Se esse comando pertencer ao usuário root um usuário qualquer irá executá-lo com as permissões do root. Por esse motivo o SUID constitui uma grande ameaça de segurança e sua utilização deve ser bastante cautelosa.

O SGID bit é geralmente atribuível a diretórios. Quando um arquivo é criado dentro de um diretório com SGID bit ativado, o conteúdo gravado dentro do diretório irá herdar o grupo do diretório e não o grupo do usuário que criou tal conteúdo. Este bit especial é muito útil quando utilizamos diretórios para grupos de trabalhos e em servidores de arquivos.

O Sitcky bit era bastante utilizado para realizar otimizações de acesso a conteúdos, entretanto, a partir da série 2.6 do kernel do Linux essa tarefa é realizada diretamente pelo kernel. A única utilidade desse bit, atualmente, é fazer diretórios de utilização comum a todos os usuários, como no /tmp.

Quando esse bit está ativo em um diretório, todo conteúdo criado dentro dele pertencerá ao criador do conteúdo e por mais que ele atribua a esse conteúdo permissões totais para todos os usuários, o único que poderá excluir o arquivo ou diretório será o próprio dono ou o root.

Para atribuirmos esses bits especiais, procedemos da mesma forma que nas permissões comuns, somando os valores e utilizando o comando chmod, mas agora utilizando quatro números, o primeiro número sendo o bit especial, seguido dos três

da permissão padrão; por exemplo:

```
 # chmod 4750 programa
```

Dessa forma as pessoas pertencentes ao grupo do arquivo poderão executar o comando 'programa' como se fossem donas desse comando. Podemos usar como exemplo o comando **passwd.**

```
 # ls -l `which passwd`
-rwsr-xr-x 1 root root 31640 2008-06-12 20:39 /usr/bin/passwd
```

Os nossos usuários comuns só podem mudar sua senha pois o comando passwd está com o bit SUID ativado. Os bits especiais são representados por um s ou S na visualização das permissões. Veja o exemplo abaixo:

```
 chmod 4000 a
chmod 2000 b
chmod 1000 c

ls -l

total 0
---S------ 1 bruno bruno 0 2008-07-21 13:50 a
------S--- 1 bruno bruno 0 2008-07-21 13:50 b
---------T 1 bruno bruno 0 2008-07-21 13:50 c
```

O bit especial para o campo de permissões de dono é o SUID representado por s ou S, para o grupo é SGUI também representado por s ou S, já o campo de permissões de outros usuários, o Sticky BIT, é representado por t ou T

Veja que quando seu determinado campo não tem permissão de execução, o bit especial é representado por uma letra S (Upper Case), e quando o campo possui uma permissão de execução o bit especial é apresentado como s (Lower case). O mesmo acontece com o Sticky bit mas com a letra t(T).

```
 chmod 4100 a
chmod 2010 b
chmod 1001 c

ls -l

total 0
---s------ 1 bruno bruno 0 2008-07-21 13:50 a
------s--- 1 bruno bruno 0 2008-07-21 13:50 b
---------t 1 bruno bruno 0 2008-07-21 13:50 c
```

Todas as permissões especias que não contiverem execução são maiúsculas. S e S e T

10.14. Prática Dirigida

1) Crie um diretório chamado temp e dentro dele um arquivo chamado arq1:

```
 $ mkdir temp
$ touch temp/arq1
```

2) Determine quais são as permissões com as quais eles foram criados:

```
 $ ls -ld temp
$ ls -l temp
```

3) Altere as permissões do diretório temp para que o dono não possa escreve-lo:

```
 $ chmod u-w temp
```

4) Experimente criar mais um arquivo dentro deste diretório:

```
$ touch temp/arq2
```

Não funcionou? Claro! Devemos alterar a permissão para que o usuário possa escrever neste diretório.

5) Mas desta vez, utilizaremos a forma numérica:

```
$ chmod 300 temp
```

6) Vamos acessar o diretório e ver se o arquivo foi realmente criado.

```
$ cd temp
$ ls -l
```

7) Temos que garantir a permissão de leitura também. Agora deverá funcionar!

```
# cd ..
# chmod 700 temp
# ls -l
```

8) Adicione um novo usuário no sistema:

```
# adduser aluno
```

9) Visualize o novo usuário no arquivo /etc/passwd:

```
# cat /etc/passwd
```

10) Logue com o novo usuário em outro terminal e veja quais usuários estão logados no sistema:

```
$ users
```

11) Logado com o novo usuário, visualize informações completas sobre ele:

```
$ finger aluno
```

12) Adicione o usuário em um grupo secundário e visualize isso:

```
# gpasswd -a aluno audio
# id
```

13) Remove o usuário do grupo secundário e visualize:

```
# gpasswd -d aluno audio
# id
```

14) Logado com root, modifique a senha do usuário:

```
# passwd aluno
```

15) Agora logado como usuário normal, modifique sua própria senha:

```
$ passwd
```

16) Modifique o grupo primário do usuário:

```
# usermod -g users aluno
```

17) Visualize qual usuário está logado no momento:

```
# w
```

18) Remova o usuário junto com o seu diretório:

```
# userdel -r aluno
```

19) Vamos agora testar a permissão especial SUID:

```
$ /sbin/ifconfig eth0:1 192.168.242.21
```

Não funcionou???

20) Vamos agora aplicar o SUID.

```
# chmod 4755 /sbin/ifconfig
$ /sbin/ifconfig eth0:1 192.168.242.21
```

Este é apenas um exemplo para fins de demonstração do SUID bit, favor não aplicar em servidores!

10.15. Exercícios Teóricos

1) Qual é a importância de se utilizar senhas sombra (shadow passwords)?

____
____
____
____

2) Por que é importante um usuário estar vinculado a um grupo primário?

____
____
____
____

3) Quais são as divisões de tipos de usuários no sistema GNU/Linux?

____
____
____

4) Por que não é aconselhável remover um usuário e manter o seu diretório home?

____

____

____

____

5) Qual é o comando correto para listar o conteúdo do arquivo /etc/passwd?

____

____

____

____

6) Considere os seguintes comandos:

```
# gpasswd -a teste1 cdrom
# usermod -G cdrom teste1
```

Explique a diferença entre eles.

____

____

____

____

7) Considere um arquivo com as seguintes permissões e os seguintes comandos:

```
 -rwxr-xr-x 1 usuario grupo 0 Dec 31 23:59 arquivo
 # chmod 644 arquivo
# chmod u+rw,g+r,o+r arquivo
```

Qual é o número da permissão original do arquivo e explique qual efeito ambos os comandos teriam sobre o arquivo, e mostre quais seriam suas permissões resultantes.

____

____

____

____

8) Se a minha umask vale 012, qual será o valor de arquivos e diretórios novos que venham a ser criados ?

____

____

____

____

10.16. Laboratório

1) Adicione um usuário chamado aula1 e visualize o conteúdo de arquivos do seu diretório home.
2) Crie um novo arquivo no diretório /etc/skel.
3) Agora adicione outro usuário chamado aula2, visualize o conteúdo de arquivos do seu diretório home e novamente o do usuário aula1.
4) Edite o /etc/passwd, modifique o UID do usuário aula1 para 0 e veja o que acontece.
5) Remova o usuário aula2 do grupo aula2.
6) Remova o usuário aula2 sem remover o seu diretório home e visualize as permissões do seu diretório home.
7) Agora adicione um usuário chamado aula3 e visualize as permissões de todos os diretórios home.

Capítulo 11

Administração da Shell

11.1. Objetivos

- Entender o funcionamento do terminal;
- Conhecer alguns tipos de shells;
- Personalizar o shell;
- Localizar os arquivos de configurações relacionados ao shell;

O principal meio de interação do usuário com um sistema GNU/Linux é o terminal de comandos, também conhecida como shell. Neste capítulo iremos aprender como personalizá-la e sobre sua utilização básica.

11.2. O que é uma shell?

A shell é uma camada de acesso ao sistema básico, o sistema operacional do computador, que pode ser acessada tanto pelo modo gráfico, quanto em modo texto. A shell pode ser praticamente 100% personalizada. As principais modificações são

com relação a sua língua padrão, personalizações de prompt e processos automáticos. Nos tópicos a seguir, veremos como fazer essa personalização.

A figura abaixo ilustra como podemos posicionar a shell dentro do sistema.
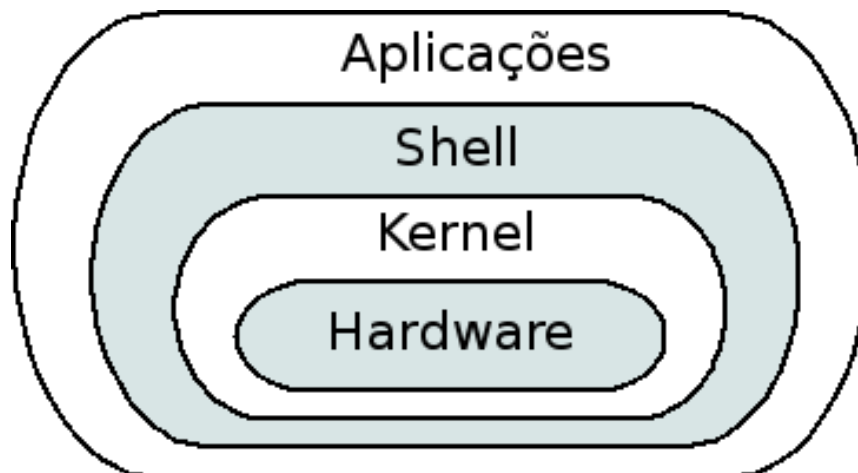
Ilustração 16: Estrutura da shell

11.3. Variáveis em Shell

As variáveis da shell têm o mesmo objetivo das variáveis que conhecemos na área da computação, ou seja, são áreas de memória que podem conter dados.

Quando estamos falando de variáveis em shell não precisamos nos preocupar em declará-las nem mesmo definir o seu tipo. Em shell, uma variável é definida simplesmente atribuindo-se um valor a ela. Vejamos um exemplo: Se definirmos que ANSWER=42, estaremos armazenando o valor 42 em um determinado endereço de memória que podemos acessar utilizando o nome que atribuímos a ele, ou seja ANSWER.

```
# ANSWER=42
```

Para acessarmos o endereço de memória atribuído à variável ANSWER, em shell devemos utilizar o operador $ (cifrão) antes do nome da variável, ou seja, se desejarmos mostrar na tela o valor da variável ANSWER devemos imprimir o

conteúdo armazenado no endereço de memória $ANSWER:

```
# echo $ANSWER
```

Esse tipo de variável que acabamos de definir é conhecida como escalar e pode receber valores numéricos ou caracteres.

11.3.1. Variáveis Locais e de Ambiente (globais)

Quando falamos em variáveis em shell temos que ter em mente a divisão entre variáveis locais de ambiente (ou globais). A diferença entre elas é que uma variável local tem visibilidade restrita apenas ao escopo ao qual ela foi definida e uma variável de ambiente tem visibilidade não só no escopo em que foi definida mas também em ambientes derivados do ambiente ao qual ela foi definida.

A única diferença entre variáveis locais e de ambiente é na hora de defini-las. Para definir uma variável local, basta atribuir um valor a um nome de variável. Para definir uma variável de ambiente o procedimento é basicamente o mesmo, adicionando o comando export antes da definição. São exemplos de definição de variável local e de ambiente os seguintes:

```
 # LOCAL="sem export na frente"
# export GLOBAL="com export na frente"
```

Uma vez definidas as variáveis podemos visualizá-las utilizando os comandos set e env para variáveis locais e de ambiente respectivamente. Com isso, se tivéssemos definido as variáveis LOCAL e GLOBAL e executássemos o comando set, veríamos as definições de ambas. Mas, se executássemos o comando env, veríamos apenas a definição da variável GLOBAL.

Variáveis de ambiente (as globais) são muito importantes pois definem a forma com que a shell e diversos outros programas irão se comportar. Por exemplo, a forma com que o prompt é apresentado ao usuário é defino pela variável global PS1.

Saber o conteúdo de algumas variáveis é muito importante, anote o dessa aqui: HISTSIZE=500

11.4. Alias

Um recurso da shell que facilita muito a vida do usuário é a definição de alias. Imagine só que um usuário gosta de utilizar o comando ls sempre com os parâmetros ``- -color -h -l''; o que seria dele se toda vez que fosse executá-lo tivesse que escrever o comando com todos os parâmetros!! Ele perderia muito tempo e acabaria se cansando mais.

Para resolver esse tipo de situação, basta criar um alias para o comando ls que defina que cada vez que o usuário digitar um simples ls o que será executado será o ls - -color -h -l. Para criarmos esse alias, basta usarmos o comando abaixo:

```
# alias ls='ls --color -h -l'
```

Dessa forma fica fácil criar um comando novo. Por exemplo, um que liste apenas diretórios:

```
# alias lsd='ls --color -h -l |grep ^d'
```

Tanto os alias quanto as definições de variáveis podem ser efetuadas em linha de comando ou, para maior comodidade, utilizando arquivos apropriados para isso.

11.4.1. Arquivos de Login

Quando uma bash é executada como uma shell de login interativo ela lê e executa o arquivo /etc/profile se ele existir. Esse arquivo deve conter as configurações gerais que se aplicam a todos os usuários do sistema.

Após ler o /etc/profile, ela irá procurar por um dos arquivo:

- ~/.bash_profile
- ~/.bash_login
- ~/.profile

, na home do usuário, executando o primeiro que estiver disponível e tiver permissão de leitura. Além desses, executa também o arquivo ~/.bashrc.

Quando a bash estiver sendo terminada (usuário fazendo logout), o arquivo ~/.bash_logout será lido e executado caso ele exista. Através deste arquivo, podemos automatizar procedimentos como por exemplo limpar a tela ao se deslogar do sistema.

Quando uma bash é chamada mas não é uma shell de login, o arquivo chamado será apenas o ~/.bashrc.

Sendo assim, se desejarmos criar alias ou definir variáveis ou funções que sejam comuns a todos os usuários, devemos fazer isso no arquivo /etc/profile. Caso o usuário não deseje utilizar o padrão do sistema, alterá-lo ou adicionar configurações pessoais, ele deve utilizar os arquivos ~/.bash_profile, ~/.bash_login, ~/.profile ou ~/.bashrc.

11.4.2. Arquivos /etc/issue e /etc/motd

Os arquivos /etc/issue e /etc/motd são usados para mostrar mensagens para os usuários e não interferem na parte operacional do sistema.

A diferença entre o os arquivos /etc/issue e /etc/motd, é que, o primeiro exibe uma mensagem para o usuário antes que o mesmo faça login no sistema, enquanto o segundo exibe uma mensagem após o usuário se logar no sistema. Há ainda o arquivo /etc/issue.net, que contém a mensagem exibida em logins remotos.

Veja um exemplo de /etc/motd do Debian:

```
#cat /etc/motd

Linux gandalf 2.6.18-4-486 #1 Wed May 9 22:23:40 UTC 2007 i686

The programs included with the Debian GNU/Linux system are free
software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
```

Veja um exemplo de /etc/issue no Debian:

```
 #cat /etc/issue
Debian GNU/Linux 4.0 \n \l
```

Os caracteres \n \l no arquivo /etc/issue representam respectivamente o nome do servidor e do terminal em que o usuário está logado.

11.5. Tipos de shell

Para saber quais shells estão disponíveis, basta visualizar o conteúdo do arquivo /etc/shells. A maioria das distribuições GNU/Linux traz a bash como shell padrão, e que é uma evolução do Bourne Shell (/bin/sh), que tem bem poucos recursos.

Para alterar o shell atual, utilizamos o comando chsh.

Exemplo:

```
 # chsh -s /bin/sh
```

No próximo login, o shell do usuário será o sh.

Conhecer um pouco das shells sh e bash pode ser de muita importância para prova.

11.6. Prática Dirigida

1) Crie uma variável local LINUX com o valor 4linux em linha de comando:

```
# LINUX=4linux
```

2) Verifique o valor desta variável:

```
# echo $LINUX
```

3) Verifique se a variável aparece na relação do comando set e do comando env:

```
# set | more
# env | more
```

4) Defina uma variável global (de ambiente) chamada GNULINUX e que receba o valor rocks:

```
# export GNULINUX=rocks
```

5) Verifique o valor desta variável:

```
# echo $GNULINUX
```

6) Agora, verifique se a variável aparece na relação dos comandos set e env:

```
# set | more
# env | more
```

7) Execute uma nova bash no mesmo terminal que você definiu as variáveis:

```
# bash
```

8) Verifique quais variáveis estão definidas:

```
# set | more
# env | more
```

9) Saia da bash filha:

```
# exit
```

10) Liste e depois altere o valor da variável de ambiente PS1:

```
# echo $PS1
# export PS1="C:\> "
```

11) Péssimo! Volte ao normal!

```
# source /etc/profile
```

12) Delete a variável LINUX da memória:

```
# unset LINUX
```

13) Crie um alias para o comando ls, de forma que as listagens sejam exibidas coloridas, mostrando o tamanho dos arquivos em formato``human readable''.

```
# alias ls='ls --color=auto -lh'
```

14) Remova o alias para o comando ls.

```
# unalias ls
```

15) Edite o seu arquivo .bashrc e adicione no final do arquivo os seguintes alias:

```
# vi ~/.bashrc
```

alias ls='ls --color=auto -lh'

alias c='clear'

alias cl='ls ; clear'

16) Teste os alias que acabamos de criar:

```
# ls
# c
# cl
```

Não funcionou? Claro, esses arquivos são lidos apenas uma vez, quando realizamos o login.

17) Faça com que os arquivos de ``inicialização'' da bash sejam relidos:

```
# source ~/.bashrc
```

18) Liste todos os alias que estão definidos:

```
# alias
# vi /etc/issue
# vi /etc/motd
# vi /etc/issue.net
```

19) Mude o shell do usuário fulano para /bin/ksh:

```
# chsh -s /bin/ksh fulano
```

11.7. Exercício Teórico

1) O que você entende por shell?

2) O que são variáveis locais?

3) O que são variáveis globais?

4) Qual a finalidade dos arquivos /etc/motd e /etc/issue?

5) Qual a finalidade dos arquivos /etc/profile e ~/.bashrc?

6) Em qual arquivo estão listados os shells disponíveis?

7) Determine quais são os conteúdos e para que servem as seguintes variáveis de ambiente: PS1, PATH, TERM, HOME, USER, PWD, OLDPWD e SHELL.

____
____
____
____

8) Considere os seguintes comandos:

```
# cd /usr/share/doc
# pwd
# cd ~
# cd -
```

Determine quais variáveis estão sendo lidas para substituir os caracteres ``~'' e ``-'' e na execução do comando ``pwd''.

____
____
____
____

11.8. Laboratório

1) A troca de sistema operacional ocorreu a menos de um mês na empresa Tab's Org, e os funcionários ainda estão um pouco confusos sobre os comandos no Linux. Para minimizar esse problema, você resolve criar novos comandos no sistema visando ajudar os usuários a se acostumarem com o shell do Linux:

1.1) alias: lista

Sua função é listar o conteúdo do diretório atual.

1.2) alias (desafio): infogeral

Sua função é mostrar algumas informações. A saída desse comando deverá ser:

- Usuário logado:
- Você está em: /etc

- Caminho dos executáveis no sistema: /usr/local/bin:/usr/bin:/bin:/usr/games
- Nome da máquina: trinity
- Data: Qua Fev 28 08:02:49 BRT 2007

2) Crie dois alias chamados lf e ldir que listem apenas arquivos, incluindo arquivos ocultos e liste apenas diretórios, incluindo diretórios ocultos.
3) Por questões de segurança, altere o conteúdo dos arquivos de mensagem de pré-login e de mensagem do dia para que contenham a seguinte instrução.

```
================================
= Este computador / sistema é de acesso restrito =
= apenas a pessoas autorizadas. Todas as suas =
= atividades estão sendo monitoradas e logadas! =
=================================
```

Capítulo 12

Compactadores, Empacotadores e Procedimentos de Backup

12.1. Objetivos

- Diferenciar compactação de empacotamento;
- Compreender o comando tar e cpio;
- Comparar as qualidades do gzip e do bzip2

A compressão e empacotamento de arquivos e diretórios é muito importante em qualquer sistema computacional. Ambos os procedimentos são necessários desde o ponto de vista de distribuição de softwares, quanto de economia de banda e de espaço de armazenamento, até backup do sistema. Veremos neste capítulo o principal programa de empacotamento GNU/Linux e os dois principais compactadores.

12.2. Empacotador TAR

A forma mais conhecida de realizar compressão e empacotamento em ambiente Windows é utilizando o programa zip. Um programa que ``zipa'' um arquivo, ou diversos arquivos, na realidade está realizando dois procedimentos distintos: Empacotar e comprimir.

Em ambientes Unix-like, essas duas tarefas são realizadas de forma logicamente distintas.

O programa tar, cujo nome deriva de ``tape archiver'', realiza a tarefa de concatenar todos os arquivos e diretórios preservando as informações do filesystem, isto é, seus meta-dados.

Criado com propósito de backup em dispositivos de acesso sequencial (unidades de fita) , o tar é utilizado hoje em dia como uma ferramenta de empacotamento, podendo ser utilizado em conjunto com compactadores como gzip ou bzip2.

A utilização da ferramenta tar é bastante simples e pode ser resumida da seguinte forma:

```
$ tar <operações/opções> f <arquivo_tar> [<arquivos_de_entrada>]
```

Onde o significado dos parâmetros são:

<operações> podem ser:

- **c -** criar o arquivo tar;
- **r -** adicionar (concatenar) conteúdo a um arquivo tar;
- **x -** extrair o conteúdo de um arquivo tar;
- **t -** listar o conteúdo de um arquivo tar.

<opções> podem ser:

- **v -** Verbose tar;
- **z -** utilizar com compressão gZip, tanto na criação como na extração de um arquivo tar;

- **j -** utilizar com compressão bzip2, tanto na criação como na extração de um arquivo tar;
- **Z -** utilizar com compressão compress, tanto na criação como na extração de um arquivo tar.

<arquivo_tar> é o nome do arquivo tar sob o qual se está trabalhando. Deve ser precedido pela letra f de file.

<arquivos_de_entrada> listagem dos arquivos que serão adicionados ao arquivo tar.

Esquematicamente, a utilização básica do comando tar pode ser ilustrada da seguinte forma:

$$\text{tar} \left| \begin{array}{c} c \\ r \\ x \\ t \end{array} \right. \begin{bmatrix} z \\ j \\ Z \end{bmatrix} [v]f \ \text{<arquivo_tar>} \ [\text{<arquivos_de_entrada>}]$$

Ilustração 17: Fluxo do Tar

Seguindo o filosofia Unix ``faça apenas uma tarefa, mas faça bem feito'' o tar é um programa especialista em empacotar vários arquivos. Dessa forma, quando utilizamos os parâmetros z e j estamos na realidade fazendo uma chamada externa aos comandos gzip e bzip2, especialistas em compressão de dados.

Podemos ver o header de um arquivo tar utilizando o comando od

```
# od -t c arquivo.tar | head -n 30
```

Outros programas que trabalham de forma análoga ao tar são o dump e cpio. Ambos foram criados com a mesma finalidade, mas são pouco utilizados hoje em dia, pois não são tão versáteis quanto o tar.

12.3. O empacotador cpio

Para empacotar com o cpio devemos fazer o seguinte.

```
# ls | cpio -ov > pacote.cpio
```

O comando acima empacotará todos os objetos da saida do comando ls.

Para extrair podemos proceder da seguinte forma:

```
# cpio -iv < pacote.cpio
```

Para compactar um diretório podemos utilizar o comando find de apoio :
 # find . teste/ | cpio -ov > a.cpio

12.4. Compactadores GZIP e BZIP2

Compressão de dados é o processo de codificar a informação de forma que seja possível armazená-la em um número menor de bits. Por exemplo, se definíssemos que a palavra ``compressão'' passaria a ser abreviada por ``comp'', estaríamos diminuindo o número de bits necessários para armazenar essa apostila.

Entretanto, para você pudesse entender o que ``comp'' significa você teria que estar ciente dessa convenção ou seja, do algoritmo de compressão.

Há dois tipos básicos de compressão, aquele em que não há perdas de informações e aquele em que elas ocorrem. Obviamente quando o assunto é backup de informações vitais devemos utilizar algoritmos sem perdas. Já em arquivos de imagens, vídeos e áudio, há casos que podemos nos dar ao luxo de perdas de informações em detrimento da qualidade, que em geral é praticamente imperceptível

para os não especialistas da área.

Os principais programas de compressão que utilizaremos são o bzip2, gzip, zip e compress. O bzip2 utiliza os algoritmos Burrows-Wheeler transform e Huffman coding; já o gzip e o zip utilizam os algoritmos LZ77 e Huffman coding; p compress utiliza o algoritmo LZW, o mesmo utilizado pelo formato de imagens gif. Todos esses algoritmos fazem parte do grupo dos algoritmos que não ocasionam perdas de dados.

A forma de utilização desses comandos são bastante simples, para o gzip, bzip2 e compress, basta fornecer o arquivo de entrada que a compressão se dará no próprio arquivo. Eis uma diferença entre o tar e esses programas. Como o programa zip realiza a tarefa de empacotar e comprimir ele recebe dois argumentos, o arquivo de saída .zip e os arquivos de entrada, ou seja, aqueles a serem empacotados e comprimidos.

Você deve estar se perguntando, se o zip já realiza o trabalho de empacotar e comprimir, para que eu utilizaria o comando tar em conjunto com um programa de compressão?!?! A resposta é simples: o zip não armazena os metadados!

12.5. Falando de Backup

Nosso capítulo explica muitas coisas sobre compactação e empacotamento de arquivos, tudo isso é extremamente necessário quando falamos de backup. Podemos ter ter tipos de backup diferentes, são eles:

Incremental O backup incremental visa salvar as diferenças em relação ao ultimo backup completo, por exemplo: Um backup completo acontece no domingo. O incremental salvará os dados de domingo para segunda, de domingo para terça, de domingo para quarta, de domingo para quinta, de domingo para sexta e de domingo para sábado, ou seja, até chegar outro backup completo.

Diferencial Diferente do incremental, o diferencial, faz apenas os incrementos, assim gerando um volume menor de dados. Se o backup completo foi gerado no domingo, ele salva de domingo para segunda, de segunda para terça, de terça para quarta e assim até o próximo backup completo chegar.

Completo - Visa salvar todos os dados, mesmo o que já foram salvos

anteriormente, assim fazendo um backup completo de todos os objetos em questão.

Conhecendo os tipos de backup, vamos conhecer algumas ferramentas que podem nos ajudar.

12.5.1. O comando dd

O comando dd tem a capacidade de copiar bit a bit uma partição, para isso podemos utilizar assim:

```
# dd if=/dev/hda /dev/hdb
```

O comando acima efetuará a clonagem do disco hda, para o disco hdb.

12.6. Prática Dirigida

1) Copie todo o conteúdo do diretório /etc para /backup e vá para o /backup:

```
# cp -rL /etc /backup
# cd /backup
```

2) Verifique que não é possível compactar um diretório sem empacotá-lo antes. Tente com o gzip e com o bzip2:

```
# gzip etc
# bzip2 etc
```

Para determinarmos qual o melhor compactador vamos analisar dois tipos de arquivos, texto puro e binário. Para isso vamos criá-los.

3) Vamos criar dois arquivos de texto puro. Abra o arquivo texto1 no editor vim e insira uma linha contento os números de 0 a 9:

```
# vim texto1
0123456789
```

4) Ainda dentro do vim, copie essa linha e cole 250.000 vezes:

```
:1y
250000p
```

5) Saia do arquivo salvando as alterações:

```
:x
```

6) Crie uma cópia deste arquivo chamando-a de texto2:

```
# cp texto1 texto2
```

7) Vamos criar um par de arquivos binários para nossos testes. Utilizaremos como base, o programa aptitude:

```
# cp /usr/bin/aptitude bin1
```

8) Vamos duplicar esse arquivo:

```
# cp bin1 bin2
```

9) Verifique que criamos quatro arquivos com tamanho parecido, aproximadamente 2.4MB, sendo dois binários e dois texto puro:

```
# ls -lh
```

12.6.1. gzip e bzip2 com arquivos de texto puro

1) Utilize a tabela tab:comparacao1 para anotar os resultados obtidos nos testes com gzip e bzip em arquivos de texto puro:

- **Tipo do Arquivo**: TEXTO puro

- **Tamanho Original:**

	GZIP	***BZIP2***
Tamanho final		
Tempo para compactar		
Tempo para descompactar		

Vamos iniciar os testes com os arquivos texto.

2) Determine o intervalo de tempo que leva para comprimir o arquivo texto1 com gzip:

```
# time gzip texto1
```

3) Determine o tamanho final do arquivo texto1 após ser comprimido com gzip:

```
# ls -lh texto1.gz
```

4) Determine o intervalo de tempo que leva para descomprimir o arquivo texto1.gz:

```
# time gunzip texto1.gz
```

Vamos repetir os procedimentos utilizando o bzip2.

5) Determine o intervalo de tempo que leva para comprimir o arquivo texto2 com bzip2:

```
# time bzip2 texto2
```

6) Determine o tamanho final do arquivo texto2 após ser comprimido com bzip2:

```
# ls -lh texto2.bz2
```

7) Determine o intervalo de tempo que leva para descomprimir o arquivo texto2.bz2:

```
# time bunzip2 texto2.bz2
```

12.6.2. gzip e bzip2 com arquivos binários

1) Utilize a tabela tab:comparacao2 para anotar os resultados obtidos nos testes com gzip e bzip em arquivos binários:

- **Tipo do Arquivo**: Binário
- **Tamanho Original:**

	GZIP	***BZIP2***
Tamanho final		
Tempo para compactar		
Tempo para descompactar		

2) Determine o intervalo de tempo que leva para comprimir o arquivo bin1 com gzip:

```
# time gzip bin1
```

3) Determine o tamanho final do arquivo bin1 após ser comprimido com gzip:

```
# ls -lh bin1.gz
```

4) Determine o intervalo de tempo que leva para descomprimir o arquivo bin1.gz:

```
# time gunzip bin1.gz
```

Vamos repetir os procedimentos utilizando o bzip2.

5) Determine o intervalo de tempo que leva para comprimir o arquivo bin2 com bzip2:

```
# time bzip2 bin2
```

6) Determine o tamanho final do arquivo bin2 após ser comprimido com bzip2:

```
# ls -lh bin2.bz2
```

7) Determine o intervalo de tempo que leva para descomprimir o arquivo bin2.bz2:

```
# time bunzip2 bin2.bz2
```

12.6.3. Trabalhando com o tar

1) Dentro do /backup vamos criar um arquivo ``tar'' que cujo conteúdo será o diretório /backup/etc empacotado. Verifique que o arquivo foi criado.

```
# tar cf etc.tar etc
# ls -lh
```

2) Liste o conteúdo do arquivo etc.tar:

```
# tar tf etc.tar
```

3) Antes de extrair o conteúdo do arquivo etc.tar, vamos renomear o diretório /backup/etc para /backup/etc.orig e verificar o conteúdo do diretório /backup:

```
# mkdir /backup/etc
# mv /backup/etc /backup/etc.orig
# ls -lh
```

4) Sendo o tar uma ferramenta de backup devemos seguir a regra número 1 de um sistema de backups... ver como se faz para restaurar um backup. Vamos extrair o arquivo etc.tar:

```
# tar xf etc.tar
```

5) Veja que um novo diretório /backup/etc foi criado a partir do arquivo etc.tar:

```
# ls -lh
```

6) Tão importante quanto conseguir restaurar um backup é que ele esteja intacto. Sendo assim, verifiquemos a integridade dos dados recuperados comparando-os com os originais:

```
# diff -r etc.orig etc
```

Se o comando diff não retornar nada na tela, significa que ambos os diretórios e seus conteúdos estão idênticos.

7) Já que estamos realizando um procedimento bastante utilizado para backup vamos comparar os espaços em disco utilizados antes e depois do backup:

```
# du -hs etc
# ls -lh etc.tar
```

Até o momento aprendemos que os compactadores gzip e bzip2 não compactam um diretório recursivamente. Sendo assim, devemos empacotar esse diretório e depois comprimi-lo utilizando algum dos programas disponíveis.

8) Crie uma cópia de segurança do diretório /etc utilizando o tar com compressão gzip:

```
# tar czf etc.tar.gz etc
```

9) Crie uma cópia de segurança do diretório /etc utilizando o tar com compressão bzip2:

```
 # tar cjf etc.tar.bz2 etc
```

10) Extraia os conteúdos dos arquivos tar.gz e tar.bz2:

```
 # tar xzf etc.tar.gz
# tar xjf etc.tar.bz2
```

12.7. Exercícios Teóricos

1) Qual a diferença entre empacotar e comprimir?

____

____

____

____

2) Quais comandos do GNU/Linux você conhece que realizam compressão?

____

____

____

____

3) Quais comandos do GNU/Linux você conhece que servem para empacotar arquivos e diretórios?

____

____

____

____

Capítulo 13

Shell Script I

13.1. Objetivos

- Entender a estrutura de um script;
- Automatizar tarefas;
- Conhecer algumas variáveis importantes;

Tarefas administrativas são, muitas vezes, longas e repetitivas. Podemos automatizar esses procedimentos através de scripts. Na verdade, os scripts podem nos auxiliar muito, numa vasta gama de atividades.

13.2. O que é um script?

Um script é uma seqüência de instruções que são executadas toda vez que o mesmo é chamado.

Mas, qual a diferença entre um script e um programa, já que ambos são seqüências de instruções?

Um script, é um programa não compilado. O processador da máquina só é capaz de executar programas binários, isto é, compilados especificamente para ele. Dessa forma, é necessário um programa que interprete esse script, em tempo de execução, para que o mesmo possa ser executado. No nosso caso, esse programa será uma shell, já que estamos falando de shell scripts.

Sendo uma linguagem de programação, a Shell Script possui uma série de estruturas de controle como loops e condicionais, mas que são estudadas apenas em cursos mais avançados. Neste curso aprenderemos a fazer um script básico.

13.3. Estudando um exemplo

Vejamos o seguinte exemplo de Shell Script:

```
1 # vim mps.sh
2 #!/bin/bash
3 #Meu primeiro shell script
4 cd ~
5 clear
6 ls -alh
7 date
8 cd -
```

Este é um script bem simples. As linhas que começam pelo símbolo # são comentários, ou seja, tudo que aparece depois do # é desprezado. Os comentários são muito importantes nos programas, pois são uma forma de documentá-los. Imagine se você tiver que fazer uma alteração num programa escrito a um ano antes. Será que você irá se lembrar de todas as estruturas e variáveis que utilizou? Provavelmente que não. Se for outra pessoa quem tiver que efetuar essa mudança, a situação será pior ainda!

Mas a primeira linha, que parece um comentário, possui uma característica um tanto estranha. Na verdade, a primeira linha de um script, quando conter em seu início a seqüência #! indica qual programa irá interpretar aquele script. No nosso exemplo será o programa /bin/bash, uma shell. Se estivéssemos criando um script Perl, a primeira linha seria algo como #!/usr/bin/perl.

Mas o script, propriamente dito, executa 4 comandos simples: Acessar o diretório do usuário corrente (cd ~); limpar a tela (clear); listar o conteúdo (ls -alh); imprimir a data (date) e voltar ao diretório original (cd -).

13.4. Executando o script

Como vimos, um programa ou script em Linux deve possuir permissão de execução. Supondo que nosso script denomina-se s1, para podermos executá-lo, devemos executar o comando:

```
# chmod u+x s1
```

E em seguida executar o script:

```
# ./s1
```

Lendo parâmetros da linha de comandos

Em algumas situações, pode ser necessário fornecer parâmetros para um script. Por exemplo, se ao invés de listar o conteúdo do diretório pessoal do usuário, quiséssemos que o script listasse o conteúdo de um diretório qualquer. Supondo que esse novo script chama-se s2, uma possível forma de utilização do script seria:

```
# ./s2 /etc
```

Para passar parâmetros para esse script, precisamos conhecer a função de algumas variáveis: $1, $2. Quando passamos algum parâmetro para o nosso script,

esse parâmetro fica armazenado em uma variável especifica. Pro exemplo:

```
 # ./script parâmetro1 parâmetro2 parâmetro3
```

Para conseguirmos resgatar o valor desses parâmetro, precisamos chamar as variáveis 1,2 e 3, por exemplo:

```
1  #!/bin/bash
2 #
3 #Esse script pega o valor dos parâmetros e imprimi na tela.
4 echo $1
5 echo $2
6 echo $3
```

```
# ./script3 42 the answer
```

13.5. Usando os números

Muitas vezes quando fazemos scripts, precisamos de uma função que faça o trabalho das operações básicas como soma,divisão,multiplicação e etc. Em shell script podemos usar o comando expr. Vamos ver esse exemplo: Um script que deve dizer quantos usuários estão presentes, quantos grupos estão presentes e no final mostrar quantos objetos meu sistema tem, a soma dos usuários e dos grupos:

```
1 #!/bin/bash
1 #
2 echo "Aguarde ....."
3 sleep 3
4 G=`wc -l /etc/group | cut -d" " -f1`
5 U=`wc -l /etc/passwd | cut -d" " -f1`
6 echo "O sistema possui $U usuários."
7 echo "O sistema possui $G grupos."
8 echo "O sistema possui `expr $G + $U` objetos."
```

13.6. Prática Dirigida

Crie o script abaixo. Dê a ele o nome userfiltro.

```
1 #!/bin/bash
2 #
3 #Esse programa tem como objetivo filtrar todos os usuários que não
possuem home
4 QTD=`cat /etc/passwd | grep -v /home/ | wc -l`
5 echo "A quantidade de usuários que não possuem home é de: $QTD"
```

Atribua a permissão de execução ao script:

```
# chmod u+x userfiltro
```

Execute o script:

```
# ./userfiltro
```

Altere o script para que ele receba um parâmetro da linha de comandos. Por exemplo o arquivo passwd.

```
1  #!/bin/bash
2 #
3 #Esse programa tem como objetivo filtrar todos os usuários que não
possuem home
4 QTD=`cat $1 | grep -v /home/ | wc -l`
5 echo "A quantidade de usuários que não possuem home é de: $QTD"
```

Execute novamente o script, fornecendo agora como argumento a localização do arquivo passwd:

```
# ./userfiltro /etc/passwd
```

Para a prova é bom saber a funcionalidade de algumas variáveis!! Miauuuuuuuu !!! $0 traz o resultado de como o seu programa foi chamado.

13.7. Usando a estrutura SE

Até o presente momento, fizemos scripts que não possuem escolhas, ou seja, comandos de execução em linha de comando em série utilizando nossa lógica para fazer com que aquele script seja executado. Mas se qualquer coisa acontecer no meio do caminho, não temos a oportunidade de trabalhar com as famosas exceptions

As Exceptions são mais conhecidas como exceções, e servem para ajudar quando o resultado de alguma parte do script pode ter vários rumos. Usando a condição se, é possível testar o resultado de uma condicional.

Por exemplo:

```
 a=1
b=2
SE $b > $a ENTÃO
    IMPRIMA $b
SENÃO
    IMPRIMA $a
FIMSE
```

13.7.1. A variável $?

A variável interrogação é conhecida por testar o valor de retorno de qualquer comando quando mostrada após sua execução. Com ela podemos verificar se o programa foi executado com sucesso ou não. Para isso basta saber que essa variável tem dois retornos principais.

```
 # comando1
# echo $?
# 0
```

Quando o resultado dessa variável é igual a 0. -------------Comando executado com sucesso!!!

```
 # comando2
# echo $?
# != 0
```

Quando o resultado é diferente de 0, quer dizer que existiu algum problema na execução do comando.

Cada programa tem sua tabela e exceções, mas sempre retornam 0 quando o programa é bem executado.

13.7.2. ***O comando test***

O teste de condicionais (strings, matemáticas e em arquivos) em Shell Script é

executado através do comando test.

Vamos conferir algumas formas de testarmos condicionais:

13.7.3. Testando strings

```
 # test "uva" = "uva"
# echo $?
# 0
```

```
# test "uva" = "banana"
# echo $?
# 1
```

13.7.4. Testando expressões matemáticas

```
 # test 5 -eq 2
# echo $?
# 1
```

```
# test 2 -eq 2
# echo $?
# 0
```

13.7.5. Testando expressões em arquivos

```
 # test -z $pinga
# echo $?
# 0
```

```
# whisk=blue
# test -z $whisk
# echo $?
# 1
```

Acima testamos algumas meios de se testar as condicionais utilizadas dentro do estrutura SE. Lembre-se que podemos usar as condicionais tanto dentro como fora da estrutura SE, denpende do caso e do meio.

Abaixo podemos ver uma lista de alguns operadores para nossa diversão.

13.7.6. Operadores de strings

Operadores	***Funções***
==	Igual
!=	Diferente

13.7.7. Operadores de matemáticos

Operadores	***Funções***
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
>	Maior
>=	Maior ou Igual
<	Menor
<=	Menor ou Igual

13.7.8. Operadores para arquivos

Operadores	***Funções***
-e	Arquivo existe
-nt	Arquivo é mais novo que

-ot	Arquio é mais antigo que
-d	É um diretório

Existem muitos outros operadores para que possamos dominar o mundo e consequentemente o sistemas UNIX om shell script.

Programar em shell script é uma arte, e como na arte, em shell o limite é a sua imaginação. Para se aprofundar nesse assunto pra lá de supimpa: http://aurelio.net/shell/canivete.html

13.8. Utilizando a estrutura if

Abaixo alguns exemplos realmente práticos de como utilizar a estrutura if.

```
1  #!/bin/bash
2 #
3 ## Primeiro script - Verificando se um usuário existe
4 #
5 echo "Digite usuário para consulta:"
6 read USER
7 REPLY=$(getent passwd | grep $USER)
8
9 if [ -z $REPLY ]; then
10  echo "Usuário $USER não existe!"
11 else
12  echo "Pagamento em dia"
13 fi
```

Ao invés de verificar se a variável que recebia o resultado do comando estava vazia, poderíamos também utilizar o comando test antes da estrutura e checar apenas seu código de erro:

```
1  #!/bin/bash
2 #
3 ## Primeiro script - Verificando se um usuário existe
4 #
5 echo "Digite usuário para consulta:"
6 read USER
7 REPLY=$(getent passwd | grep $USER)
8
9 test -z $REPLY
10 if [ $? -eq 0 ]; then
11  echo "Usuário $USER não existe!"
12 else
13  echo "Pagamento em dia"
14 fi
```

Podemos fazer diversas operações no nosso sistema com o shell. Vamos para o ultimo exemplo:

```
1 #!/bin/bash
2 #
3 ## segundo script - Verificando se um arquivo está vazio
4 #
5 # Programa para gerar uma lista de usuários com seus determinados
homes,
6 # Exportar para um arquivo com um cabeçalho com usuário e timestamp
7 # Fazer check de arquivo existente
8 echo "Deseja prosseguir com a exportação da lista de usuários? (y/n)"
9 read OP
10
11 if [ $OP = y ]; then
12    echo "Nome da lista"
13    read LISTA
14    test -e $LISTA
15    if [ $? -eq 0 ]; then
16     echo "Arquivo existe, impossível continuar .... saindo"
17    exit
18  else
19     echo "Gerando lista ...."
20     echo "Colocando o cabeçalho ..."
21     echo "Lista de usuários xpto" >> $LISTA
22     echo "Verificando usuários ..."
23     echo "Adicionando a lista ... "
24     echo "Emitida por:$USER" >> $LISTA
25     echo $(date +"Emitido dia %d de %B de %Y as %H:%M")  >> $LISTA
26     echo " " >> $LISTA
27     getent passwd | cut -d: -f1,6 | sed
    s/:/\----------------------------\>/ >> /tmp/$0.list
28     cat $LISTA /tmp/$0.list > /tmp/$0.final
29     echo " " > $LISTA
30     cat /tmp/$0.final > $LISTA
```

```
31     echo "Exibindo lista, sua lista gerada chama $LISTA"
32     cat $LISTA
33     echo Done.
34     fi
35    else
36          if [ $OP = n ]; then
37          exit
38          else
39          echo "Opção inválida"
40     fi
41
42 fi
```

Podemos explorar ainda mais o Shell Script ... Aproveite também nosso curso de Shell Script com o Julio César Neves.

13.9. Exercícios Teóricos

1) O que é um programa compilado? O que é um compilador?

____

____

____

____

2) Dê exemplos de linguagens de programação, normalmente compiladas.

____

____

____

____

3) Como são chamados os programas interpretados? De alguns exemplos de linguagens típicas.

____
____
____
____

4) O que é necessário para executar um programa não compilado, isto é, um script?

____
____
____
____

5) Compare as vantagens e desvantagens dos programas compilados e scripts.

____
____
____
____

6) Qual a função do comando test?

____
____
____
____

7) Qual operador podemo usar para comparar se um objeto é um dispositivo de bloco (man bash)?

____
____
____
____

13.10. Laboratório

1) Crie um script shell que mostre uma porção de um arquivo, onde os argumentos seriam:

- Nome do arquivo
- Número da primeira linha mostrada

- Número da última linha mostrada

2) Exemplo de uso, supondo o nome do script miolo:

```
# miolo /etc/passwd 7 18
```

3) Como resultado, as linhas 7 a 18(inclusive) do arquivo /etc/passwd devem ser mostradas.

Dica 1: Use os comandos head e tail.

Você sabe realizar aritmética de inteiros na shell?

4) Crie um script com as seguintes características:

- Front-end Simples para o comando date;
- O script deve receber uma data no formato dd/mm/yy;
- O script deve receber uma hora no formato HH:MM;
- Após receber esses dados os script deve filtrar as informações e alterar o horário do sistema.
- Com o script já funcional, adicione ao $PATH;

Capítulo 14

Agendamento de Tarefas

14.1. Objetivos

- Entender como funciona o agendamento no sistema;
- Aprender o funcionamento do cron;
- Aprender o funcionamento do at;

14.2. Introdução Teórica

A crontab é utilizada para agendar comandos para serem executados periodicamente, ao contrário do comando at, que executa comandos pontualmente. Há dois tipos de crontab a de usuários e a do sistema. Ambas são arquivos que contêm tabelas com informação de quando o comando especificado deve ser executado, sendo que cada linha corresponde a um único agendamento.

A crontab é gerenciada pelo daemon crond, que a cada um minuto verifica se há algum agendamento que deve ser executado e executa-o. A crontab dos usuários pode ser acessada pelo comando:

```
# crontab [-e|-r|-l]
```

E fica armazenada em arquivos com o nome do usuário dono da tabela no diretório /var/spool/cron/cronjobs. Já a crontab do sistema é encontrada no /etc/crontab e já possui agendamentos para realizar as tarefas que se encontram nos diretórios /etc/cron.[hourly|daily|weekly|monthly]. Sendo que o programa chamado run-parts é quem executa os referidos agendamentos.

O formato das crontabs dos usuários e do sistema são quase iguais, Ã exceção que a crontab do sistema possui um campo a mais, como pode ser visto a seguir:

```
 crontab (usuários)
# minuto hora dia mês diaDaSemana comando

crontab (sistema)
# minuto hora dia mês diaDaSemana USUÁRIO comando
```

A única diferença entre as duas crontabs é que na do sistema há um campo para especificar qual é o usuário que irá executar o comando agendado.

Além disso cada campo possui um conjunto de valores válidos, sendo eles:

- **minuto** varia de 0-59;
- **hora** varia de 0-23;
- **dia** varia de 1-31;
- **mes** varia de 1-12;
- **diaDaSemana** varia de 0-7, sendo 0 e 7 são o domingo;
- **usuário** um usuário válido no sistema;
- **comando** o path completo para o comando.

Podemos controlar quais usuários podem acessar ou não o cron, basta criar um dos arquivos, /etc/cron.allow ou /etc/cron.deny. A mesma dica é válida para o at. /etc/at.allow ou at.deny

Considerando o formato já listado, podemos realizar agendamentos utilizando alguns operadores que facilitam, como:

- **vírgula (,)' -** especifica uma lista de valores, por exemplo: ``1,3,4,7,8'';
- **hifen (-)** especifica um intervalo de valores, por exemplo: 1-15 (de 1 a 15);
- **asterisco (*) -** especifica todos os valores possíveis;
- **barra (/) -**especifica ``pulos'' de valores, por exemplo: se no campo hora utilizarmos ``*/3'' o comando será executado as ``0,3,6,9,12,15,18,21'';

Conhecendo a sintaxe básica das crontabs passemos aos agendamentos.

14.3. Prática Dirigida

14.3.1. Agendamento de Tarefas com AT

1) Verifique se a data e a hora do sistema estão corretas:

```
# date
```

Após essa verificação podemos começar a realizar agendamentos.

2) Agende para 10 minutos no futuro a listagem do diretório /etc redirecionando para um outro arquivo, e depois outro arquivo contendo a data e a hora em que foi executada:

```
 # at HH:mm MM/DD/YYYY
at> ls --color /etc > /root/etc_list.txt
at> echo $(date +"%H:%m %M/%d/%Y" ) >> /root/gera.txt
at> (Ctrl + d)
```

3) Agendada esta tarefa, confirme o agendamento listando todos os agendamentos pendentes:

```
 # atq
```

4) Vamos explorar o diretório onde ficam os agendamentos:

```
 # cd /var/spool/cron/atjobs
# ls -la
```

5) Mostre o conteúdo dos arquivos contidos nesse diretório:

```
 # cat (agendamento)
```

Repare que no agendamento, temos nossa variáveis, e o comando.

6) Vamos realizar outro agendamento, para executar em 15 minutos, para que possamos aprender como apagá-lo:

```
 # at HH:mm MM/DD/YYYY
at> echo "Teste" > /tmp/at.out
at> ^d
```

Liste os agendamentos correntes e verifique que um novo arquivo foi criado no diretório de spool do at.

7) Remova o último agendamento:

```
 # atrm <numero_agendamento>
```

Liste os agendamentos ativos e liste o conteúdo do diretório de spool do at e

veja que o job foi removido, utilize seus conhecimentos.

14.3.2. Agendando Tarefas com o CRON

8) Crie um script para fazer um backup do /etc:

```
 # vi /root/backup.sh
#!/bin/bash
# Backup do /etc

tar czf /backup/$(date +%Y%m%d-%H%M)-etc.tar.gz /etc
```

OBS: Não esqueça das permissões do script.

9) Entre na crontab do usuário para editá-la:

```
 # crontab -e
```

10) Coloque na crontab do usuário um agendamento para fazer backup do diretório /etc/ utilizando o script criado a cada 2 minutos.

```
 # MM HH DD mm DS CMD
 */2 * * * * /root/backup.sh
```

11) Visualize os agendamentos feitos pelo o usuário.

```
 # crontab -l
```

12) Onde ficam armazenados os agendamentos feitos pelos usuários com o crontab -e?

```
 # cd /var/spool/cron/crontabs
# ls
```

OBS: Atenção! Não apague ou edite o seu agendamento dentro desse diretório, use os comandos para fazer isso.

13) Após verificar que os agendamentos foram efetuados corretamente, apague todos os agendamentos do usuário.

```
 # crontab -r
```

OBS: Para apagar somente um agendamento do usuário, use o crontab -e e retire a linha desejada.

Agora que aprendemos a utilizar a crontab do usuário podemos usar a crontab do sistema que opera praticamente da mesma forma, apenas tem um campo a mais, o usuário que executará o script.

14) Faça o mesmo agendamento para execução do script backup.sh, mas agora na crontab do sistema:

```
 # vi /etc/crontab
# MM HH DD mm DS USER CMD
 */2 * * * * root /root/backup.sh
```

Repare que dentro do arquivo /etc/crontab existem quatro agendamentos já definidos: cron.hourly, cron.daily, cron.weekly e cron.monthly.

15) Para fazer com que o script backup.sh seja executado diariamente:

```
# cp /root/backup.sh /etc/cron.daily/backups
```

16) Para fazer com que o script backup.sh seja executado semanalmente:

```
# cp /root/backup.sh /etc/cron.weekly/backups
```

17) Depois de adicionar o script dentro dos diretórios será necessário reiniciar o daemon do cron.

```
# /etc/init.d/cron stop
# /etc/init.d/cron start
```

14.4. Exercícios Teóricos

1) Qual a diferença entre os agenda-dores de tarefas at e cron?

____

____

____

____

2) Quantos campos há na crontab do sistema? E na crontab do usuário?

____

____

____

____

3) Qual é a sintaxe correta para agendar uma tarefa que seja executada nos dias 1, 7, 10 e dos dias 15 ao 20 Ã s 19hs utilizando a crontab do usuário, sendo o nome do comando ``foo''?

____

____

____

____

14.5. Laboratório

1) Realize o agendamento da execução do script de backup (backup.sh) para que rode a cada 5minutos no dia de hoje, nas próximas 2hs, utilizando a crontab do sistema.
2) Crie um script, chamado uso.sh, que lista as informações de utilização de espaço pelos file systems, jogando a saída em um arquivo chamado uso.txt no /root e coloque-o no diretório apropriado para que ele seja executado junto com as tarefas diárias da crontab do sistema.
3) Manipule a data e hora do sistema para que as tarefas diárias sejam executadas daqui três minutos. Verifique se o script ``uso.sh'' foi executado.
4) Determine o motivo pelo qual o script ``uso.sh'' não foi executado.
5) Remova todos os agendamentos que realizou para não lotar o /backup. :-)

Configure o timezone no RedHat utilizando o tzselect

Capítulo 15

Instalando, removendo e atualizando programas

15.1. Objetivos

Os diversos programas GNU/Linux são distribuídos em forma de pacotes, específicos para cada distribuição. Neste capítulo aprenderemos um pouco sobre esses pacotes e como instalá-los e removê-los do sistema.

15.2. O que é um pacote?

Pacotes são conjuntos de arquivos necessários Ã execução de um software agrupados de forma a facilitar a instalação e distribuição do programa. Eventualmente podem conter sistemas de listagem/checagem de dependências, scripts para configuração.

Os pacotes nos sistemas baseados em Debian têm uma extensão característica: ***.deb.***

Já nas distribuições baseadas em RedHat, temos pacotes com a extensão característica: ***.rpm.***

15.3. Mas o que é um gerenciador de pacotes?

Um gerenciador de pacotes é um sistema de gerenciamento para a instalação, atualização e a remoção dos pacotes anteriormente instalados. Parece muito simples falar em instalação de pacotes, mas temos que lembrar que é o gerenciador de pacotes é quem faz toda a parte suja para nós.

Um pacote nem sempre depende apenas dele mesmo, ou seja, quando instalamos um programa, ele depende de bibliotecas de áudio, vídeo, imagens, funções e vários outros tipos variados de dependências.

O trabalho feito pelo gerenciador de pacotes é interpretar a necessidade de cada um dos pacotes para que eles possam funcionar de forma devida. Para os sistemas baseados em Debian, a ferramenta a ser utilizada é o **aptitude.**

Já para sistemas baseados em RedHat temos a ferramenta yum.

Nosso curso está baseado na ferramenta aptitude.

```
aptitude
```

Vamos primeiramente utilizar essa ferramenta em seu modo texto, e logo após em seu modo visual.

Para descobrirmos o nome correto do pacote que desejamos instalar, podemos fazer uma busca pelo comando abaixo:

```
aptitude search <argumento>
```

Para buscar uma lista completa de pacotes disponíveis para debian acesse: http://packages.debian.org

No RedHat podemos aplicar uma consulta executando yum search<argumento>

Voltando a falar de dependências e pacotes, temos que entender algumas coisas sobre pacotes. Os pacotes não são apenas binários mágicos que depois de um comando de instalação estão prontinhos para funcionar.

A instalação de um pacote depende de vários pré-requisitos que o próprio pacote é capaz de fornecer. Por exemplo, queremos instalar o pacote pidgin, um aplicativo de mensagens instantâneas, após comunicarmos que queremos instalar esse pacote, o nosso gerenciador de pacotes irá verificar algumas coisas importantes em relação aquele pacote, como dependências, recomendações, conflitos ou apenas sugestões.

Para poder ver essas informações podemos executar o comando:

```
aptitude show <pacote>
```

Algumas informações importantes também podem ser obtidas comando apt-cache.

As dependências são pacotes que estão diretamente ligados ao pacote que irá ser instalado, ou seja, essenciais. Se um pacote depende de outro, ambos devem ser instalados pois o programa em questão só irá funcionar se todas suas dependências estiverem supridas.

As recomendações são pacotes que não são essenciais, porém tiram alguma função que o programa poderia ter. Por exemplo, quando instalamos o pacote mozilla-browser é recomendado também a instalação do pacote mozilla-psm, que dá

suporte as paginas seguras.

As sugestões são pacotes que são relacionados a complemento de funcionalidade, a instalação desse pacote pode fornecer alguns complementos em relação ao pacote que está sendo instalado.

Os conflitos são pacotes que não podem ser instalado juntos no sistema.

15.4. Gerenciamento de pacotes

Em distribuições baseadas em Debian, a maneira mais simples de gerenciar os pacotes de software é usando o comando aptitude. O aptitude é um front-end para o sistema APT.

A primeira etapa é usar o comando:

```
# aptitude update
```

A saída do comando será algo similar a listagem abaixo. A localização dos repositórios poderá ser diferente, conforme sua configuração:

```
 Obter:1 http://security.debian.org etch/updates Release.gpg [189B]
Obter:2 http://security.debian.org etch/updates Release [22,5kB]
Obter:3 http://ftp.us.debian.org etch Release.gpg [378B]
Atingido http://ftp.us.debian.org etch Release
Atingido http://ftp.us.debian.org etch/main Packages
Atingido http://ftp.us.debian.org etch/contrib Packages
Atingido http://ftp.us.debian.org etch/non-free Packages
Obter:4 http://security.debian.org etch/updates/main Packages [90,8kB]
Atingido http://ftp.us.debian.org etch/main Sources
Atingido http://ftp.us.debian.org etch/contrib Sources
Atingido http://ftp.us.debian.org etch/non-free Sources
Atingido http://security.debian.org etch/updates/contrib Packages
Atingido http://security.debian.org etch/updates/non-free Packages
Obter:5 http://security.debian.org etch/updates/main Sources [13,3kB]
Atingido http://security.debian.org etch/updates/contrib Sources
Atingido http://security.debian.org etch/updates/non-free Sources
Baixados 127kB em 3s (34,2kB/s)
Lendo lista de pacotes... Pronto
```

O comando acima sincroniza a lista de pacotes disponíveis para instalação nos servidores remotos com a lista de disponíveis local. Existem mais de 14.000 pacotes de software disponíveis.

15.5. Espelhos e o arquivo /etc/apt/sources.list

Este arquivo contém os locais onde o APT encontrará os pacotes, a versão da distribuição que será verificada (stable, testing, unstable) e a seção que será copiada (main, non-free, contrib, non-US). Essas definições são usadas em sistema Debian.

Segue um exemplo de arquivo de configuração.

```
 # vi /etc/apt/sources.list

# Linhas que começam por # são comentários
# Repositório a partir de CD/DVD
# deb cdrom:[Debian GNU/Linux 4.0 - CD Binary-1]/ etch contrib main

# Repositórios oficiais
deb http://ftp.us.debian.org/debian/ etch main contrib non-free
deb-src http://ftp.us.debian.org/debian/ etch main contrib non-free

# Repositórios oficiais de atualizações de segurança
deb http://security.debian.org/ etch/updates main contrib non-free
deb-src http://security.debian.org/ etch/updates main contrib non-free
```

Após fazer as configurações será necessário fazer um update.

```
 # aptitude update
```

15.6. Instalação, Remoção e Atualização

Para a instalação deve-se usar o comando aptitude com a instrução install e fornecer o nome do pacote desejado. Por exemplo, para instalar o programa de navegação em linha de comando lynx, digitamos:

```
 # aptitude install lynx
```

Para remover um pacote instalado deve-se usar o comando aptitude com a instrução remove e fornecer o nome do pacote desejado. Por exemplo, para remover o programa de navegação em linha de comando lynx, digitamos:

```
# aptitude remove lynx
```

15.7. Consultas de Pacotes

Imagine que necessite de um pacote que trabalhe com arquivos JPG. O aptitude pode ser usado para consultar a base de pacotes disponíveis relacionados ao texto jpg.

```
# aptitude search jpg
```

Dessa forma, o comando irá retornar todos os pacotes que fizerem alguma relação com arquivos jpg.

15.8. Atualização via Internet

O sistema pode ser atualizado de tempos em tempos ou por questões de segurança. Para instalar todas as atualizações disponíveis, usa-se o aptitude com a instrução safe-upgrade. Dependendo da velocidade de conexão, este processo pode levar bastante tempo.

```
# aptitude safe-upgrade
```

15.9. Gerenciamento de pacotes em distros baseadas em rpm.

Nas distros baseadas em Red Hat, o gerenciamento de pacotes é feito pelo programa rpm. A Red Hat e Fedora disponibilizam também a ferramenta yum, similar em funcionalidade ao aptitude. Já o SUSE apresenta a

ferramenta zypper, muito embora nesta distro recomenda-se a utilização da ferramenta Yast para gerenciamento de pacotes e configuração do sistema. Quando falamos de mandriva a ferramenta da vez é o urpmi.

Obtendo informações:

Usando o yum para mostrar informações de pacotes

```
# yum info pacote
```

Você pode usar o yum também para buscar pacotes nos repositórios. Todos os pacotes que contém o padrão no nome serão mostrados:

```
# yum search padrão
```

15.9.1. Instalando pacotes:

Para instalar um pacote diretamente do repositório:

```
# yum install pacote
```

15.9.2. Removendo pacotes:

```
 # yum remove pacote
```

Alem do yum, outro bom gerenciados de meta-pacotes é o urpm

Estes são os comandos básicos e principais do yum. Para saber mais, consulte as man pages desses aplicativos.

Dica LPI: As questões da LPI sobre dpkg, aptitude e rpm têm peso elevado. Conheça bem estes comandos fazendo o curso 451 da Formação 4Linux

15.10. Exercício Teórico

1) O sistema APT é responsável por qual função no sistema?

___

___

___

___

2) Qual a função do aptitude update?

___

___

___

___

3) Qual a função do arquivo /etc/apt/sources.list?

___

___

___

___

4) Onde estão os arquivos temporários que o aptitude armazena para instalar?

___

___

___

___

5) Em caso de falta de espaço em disco por instalar muitos pacotes, o que você faria?

___

___

___

___

6) Quais as vantagens e desvantagens de ser ter um pacote instalado pelo código fonte?

____

____

____

____

7) Supondo que seu mirror default esteja inoperante, como você contornaria a situação?

____

____

____

____

Capítulo 16

Servidor X

16.1. Objetivos

- Configurar a interface gráfica para os usuários comuns;
- Conhecer o arquivo de configuração;
- Instalar e configurar os Display Managers;
- Instalar e configurar os Window Managers;
- Gerenciar protocolo de rede, utilizando recursos do servidor;
- Abrir novas instâncias de janelas de máquinas em rede.

16.2. Introdução Teórica

O X Window System, conhecido também como servidor X, apenas X ou X11, é um protocolo de rede e vídeo que provê a capacidade de se trabalhar com o sistema de janelas e que permite as interações através de teclado e mouse. Esse sistema fornece os meios para o desenvolvimento de interfaces gráficas para usuários ou GUI - ``Graphical User Interfaces'' em sistemas Unix e Unix-like.

O sistema X fornece apenas as ferramentas que possibilitam o desenvolvimento de ambientes GUI como desenhar na tela, mover janelas e interagir com o mouse e teclado; ele não dita quais serão as decorações das janelas, quem faz isso são os chamados Window Managers (WM) ou gerenciadores de janelas. Dessa forma, a ``cara'' da parte gráfica varia drasticamente de um WM para outro.

Um conceito básico do servidor X é que ele é realmente um servidor como o próprio nome já indica. Sendo assim, é possível abrir várias instâncias de interface gráfica em uma mesma máquina ou até mesmo em uma máquina remota, graças ao seu protocolo de rede.

16.3. Configurando o suporte à Interface Gráfica

A interface gráfica mais utilizada em ambientes UNIX é conhecida como X Window System ou simplesmente X. Essa interface é provida pelo pacote Xorg, que podem ser baixados diretamente nos site oficial http://www.xorg.org ou utilizando o "aptitude" dos pacotes necessários.

Há basicamente quatro formas de configurar o servidor X, sendo elas:

```
Automaticamente:
# dexconfig
 Ou manualmente:
 # X -configure
```

No Debian Lenny 5.0, o X tem uma configuração um pouco menor dado o fato que todas as configurações do debconf são aproveitas para configuração do servidor X;

O arquivo de configuração do servidor X é dividido em seções e cada uma diz respeito à configuração de um determinado pedaço do sistema como um todo. A estrutura básica de um desses arquivos é a seguinte:

- ServerLayout
    - InputDevice (mouse)
    - Screen
    - InputDevice (keyboard)
- Files
- Modules
- InputDevice (mouse)
- InputDevice (keyboard)
- Screen
    - Monitor
    - Displays
    - Device (video card)
- Monitor
- Device (video card)

Ou seja, o arquivo é composto de várias seções que definem qual será o comportamento dos dispositivos como teclado, mouse, monitor e placa de vídeo e algumas outras definem recursos que o servido X irá utilizar, como os módulos que serão carregados e arquivos de fontes, por exemplo.

Além das seções separadas que definem o comportamento de algum componente em separado, há outras como ``ServerLayout'' e ``Screen'' que definem como o conjunto de recursos irá operar.

16.4. Variável de Ambiente DISPLAY

A variável de ambiente DISPLAY é a que define em que lugar a saída gráfica deve ser mostrada. Com essa variável definida é possível até informar ao sistema que a saída gráfica se dará em outro computador na rede. O formato de definição dessa variável é o seguinte:

<ip_destino>:<display>.<screen>

sendo o <ip_destino> o endereço IP de uma máquina na rede, podendo ser deixado em branco caso a máquina de destino seja a própria máquina local. O campo display refere-se a uma instância de parte gráfica dentro de uma screen; o campo screen refere-se ao monitor e à placa de vídeo que irão sair a parte gráfica.

Não se esqueça que a variável que define o ambiente do usuario é a DISPLAY.

16.5. Window Managers

Um X window manager é um software que controla basicamente o posicionamento e a aparência das janelas dentro do sistema X Window.

Ao contrário dos sistemas da Apple e Microsoft, que possuem apenas uma única aparência básica e que é de controle delas, nos sistemas GNU/Linux você é livre para escolher qual é o gerenciador de janelas que irá utilizar.

Há um número muito grande de gerenciadores de janelas que você pode instalar simultaneamente em uma máquina, possibilitando que cada usuário escolha aquele que mais lhe agrade. Cada gerenciador difere do outro em muitos aspectos, como nível de customização da aparência e funcionalidades, configurabilidade dos menus, meios gráficos para iniciar um software, capacidade de utilizar múltiplos desktops e, principalmente, na quantidade de recursos que ele exige da máquina,

entre outros.

Algumas das opções de gerenciadores são:

AfterSteps	Blakbox	FluxBox
Evilwn	Enlightenment	FVWM
IceWM	Ion	Kwin(KDE)
Metacity (Gnome)	WMN	SawFish
twm	xfce	OpenClasses(Sun)

16.6. Display Managers

Os Display Managers são programas que agrupam algumas tarefas como realizar o loging do usuário local ou remoto (via protocolo XDMCP), além de permitir que o usuário selecione de forma fácil qual Window Manager ele irá utilizar.

Alguns exemplos de Display Managers são o KDM (padrão do KDE), GDM (padrão do GNOME), XDM (padrão do servidor X).

16.7. Protocolo XDMCP

O XDMCP ou X Display Manager Control Protocol é um protocolo de rede que utiliza a porta 177/udp e é utilizado para servir interface gráfica para clientes na rede.

Se um Display Manager estiver com o protocolo XDMCP ativado, basta um servidor X enviar um pacote de ``query'' à máquina que está servindo o DM que responderá à máquina solicitante enviando a saída gráfica do DM para que algum usuário realize o login.

Esta é uma forma de utilizar a parte gráfica em uma máquina com menos recursos de hardware uma vez que o processamento de interface gráfica estará ocorrendo na máquina servidora.

16.8. Xnest

Um Xnest é uma instância do servidor X que pode ser utilizada para receber alguma saída gráfica que tenha sido redirecionada a ela utilizando a variável display. Pode ser utilizada também para receber um DM solicitado via XDMCP.

16.9. Prática Dirigida

16.9.1. Instalação e Configuração do Servidor X

Até a versão Sarge do Debian, o servidor X11 utilizado era o XFree86, a partir da versão Etch, o servidor padrão passou a ser o XOrg.

1) Sendo assim, para instalar o servidor X na versão Sarge do Debian devemos utilizar o pacote x-window-system e, para o Etch, devemos utilizar o xserver-xorg.

```
# aptitude install xserver-xorg
```

Após a instalação, vamos testar se a configuração padrão serve para a nossa máquina.

2) Inicie o servidor X:

```
# X
```

Qual comando eu consigo trazer informações sobre as cores e opções do Servidor X? R: xwininfo

3) Gere a configuração de vídeo detectada pelo debconf:

```
# dexconf
```

4) Caso esteja funcionando, ótimo. De qualquer forma, vamos executar o procedimento de configuração:

```
# X -configure
```

Esse comando irá tentar identificar qual é o hardware da sua máquina e gerar um arquivo de configuração para ela gravando esse arquivo no diretório do root.

5) Teste esse novo arquivo de configuração:

```
# X -config /root/xorg.conf.new
```

Para configurar o Xorg, via dpkg, digite:

```
# dpkg-reconfigure xserver-xorg
```

6) Novamente, se funcionar, ótimo, caso não funcione, teremos que realizar os ajustes manualmente e, para isso, precisaremos de algumas informações como:

- **placa de vídeo -** para determinar qual é a nossa placa de vídeo podemos utilizar o comando:

```
lspci | grep -i VGA
```

- **frequências do monitor -** para descobrir quais são as frequências do seu monitor você deve recorrer ao manual e ao Google.

7) Vamos visualizar o arquivo de configuração:

```
# cat /root/xorg.conf
```

Um arquivo de configuração típico:

```
 Section "Files" (Fontes do meu servidor X, cuidado, ela pode ser
cobrada naLPI)
 FontPath "/usr/share/fonts/X11/misc"
 .
 .
 .
 FontPath "/var/lib/defoma/x-ttcidfont-conf.d/dirs/TrueType"
EndSection

Section "Module" (Seção responsável pelo Módulos da minha máquina)
 Load "i2c"
 .
 .
 .
 Load "vbe"
EndSection

Section "InputDevice" (Entrada de Teclado)
 Identifier "Generic Keyboard"
 Driver "kbd"
 Option "CoreKeyboard"
 Option "XkbRules" "xorg"
 Option "XkbModel" "abnt2"
 Option "XkbLayout""br"
 Option "XkbVariant" "abnt2"
EndSection

Section "InputDevice" (Entrada de Mouse)
 Identifier "Configured Mouse"
 Driver "mouse"
 Option "CorePointer"
 Option "Device" "/dev/input/mice"
 Option "Protocol" "ImPS/2"
 Option "Emulate3Buttons" "true"
EndSection

Section "Device" (Seção que define o nosso hardware de vídeo)
```

```
 Identifier "Video Card"
 Driver "vmware"
EndSection

Section "Monitor" (Opções de Monitor)
 Identifier "Generic Monitor"
 Option "DPMS"
 HorizSync 28-51
 VertRefresh43-60
EndSection

Section "Screen" (Layout de Screen, bits de cores a serem utilizadas)
 Identifier "Default Screen"
 Device "Video Card"
 Monitor "Generic Monitor"
 DefaultDepth 24
 SubSection "Display"
 Depth 1
 Modes "1024x768" "800x600" "640x480"
 EndSubSection
 .
 .
 .
 SubSection "Display"
Depth 24
Modes "1024x768" "800x600" "640x480"
EndSubSection
EndSection

Section "ServerLayout"
 Identifier "Default Layout"
 Screen "Default Screen"
 InputDevice"Generic Keyboard"
 InputDevice"Configured Mouse"
EndSection
```

Leitura sugerida: man xorg.conf para mais informações a respeito desse arquivo e suas opções de configuração e parâmetros.

Realizadas as alterações, vamos realizar um novo teste para ver se o servidor consegue subir.

8) Teste as configurações:

```
# X -config /root/xorg.conf.new
```

Se as configurações não funcionarem teremos que ler a mensagem de erro e tentar identificar o problema.

9) Funcionando, basta mover os arquivos para o diretório correto:

```
# mv /root/xorg.con.new /etc/X11/xorg.conf
```

10) Tente os seguintes comandos e diga qual é a diferença entre eles:

```
# X
# startx
```

16.9.2. Instalando um Window Manager

No Linux podemos ter vários Clientes Gráficos. Depois que o servidor gráfico já está instalado e configurado, só vamos ter o trabalho de instalar os clientes gráficos.

1) Instalar o gerenciador de janelas WindowMaker:

```
# aptitude install wmaker
```

2) Agora vamos iniciar o nosso cliente gráfico que acabamos de instalar:

```
# startx
```

3) Para um próximo teste, vamos instalar outro cliente gráfico que é muito utilizado, o KDE:

```
# aptitude install kdebase
```

4) Depois vamos iniciar nosso outro gerenciador de janelas:

```
# startx
```

Note que foi utilizado o mesmo comando para iniciar tanto WindowMaker quanto o KDE (o startx). Isso acontece porque ao instalarmos o KDE ele se colocou como sendo o WM padrão do sistema, mas isso pode ser alterado.

5) Podemos editar o arquivo /root/.xinitrc para escolhermos qual cliente gráfico será iniciado quando o root utilizar o comando startx. Essa configuração é vállida apenas para o usuário root, pois alteramos o xinitrc da home do root:

```
# vi /root/.xinitrc
```

- WindowMaker utilize: **wmaker.**
- KDE utilize: **startkde.**

6) Para que alteração seja válida para qualquer usuário, devemos editar o arquivo de configuração global:

```
# vi /etc/X11/xinit/xinitrc
```

7) Lembrando que uma configuração local, ou seja, o arquivo pessoal do usuário, prevalece sobre o global, caso o usuário especifique um. Vamos

deixar instalados os pacotes do GNOME e do XFCE

```
# aptitude install gnome xfce4
```

16.9.3. Display Managers

Vimos no tópico anterior como iniciar o nosso cliente gráfico utilizando o comando startx, mas isso nem sempre é muito prático. Para facilitar esse processo, podemo utilizar os chamados Display Managers.

1) O gerenciador padrão do Xorg é o xdm que já está instalado. Vamos iniciá-lo:

```
# /etc/init.d/xdm start
```

2) Vamos instalar o kdm, que possui mais recursos:

```
# aptitude install kdm
```

Serão feitas algumas perguntas sobre qual será o seu Display Manager Default, o kdm ou xdm; escolha sempre o kdm, pois dessa maneira toda vez que o seu sistema iniciar, ele vai levantar automaticamente o kdm no terminal 7 por padrão.

3) Para iniciar o kdm é da mesma maneira. Lembrando que o xdm deve estar parado!

```
# /etc/init.d/xdm stop
# /etc/init.d/kdm start
```

4) Se quiser mudar o seu display manager default, basta editar o seguinte arquivo:

```
 # vi /etc/X11/default-display-manager
/usr/bin/kdm
```

5) Por fim, vamos conhecer outro DM, o GDM, padrão do GNOME:

```
 # aptitude install gdm
```

O Window Manager Padrão é GNOME.

No Xorg o arquivo de configuração: /etc/X11/xorg.conf. Peso elevado

16.9.4. Usando o Xnest

Objetivo:

Queremos rodar um aplicativo na nossa máquina local mas que esteja sendo executado em uma máquina remota. Uma ilustração dos procedimentos pode ser vista na figura fig:xnest. Sendo assim:
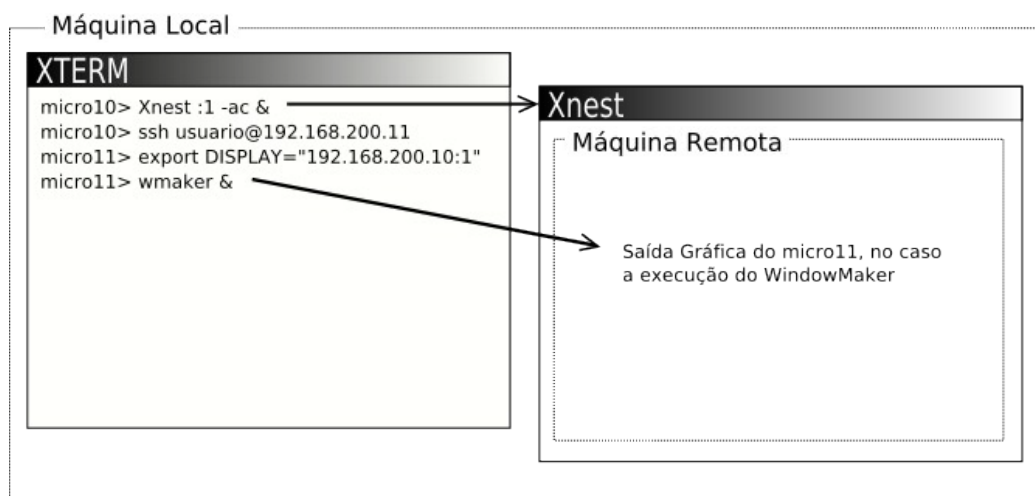
Ilustração 18: xnest

1) Vamos iniciar o nosso Window Manager utilizando o Display Manager

KDM. Faça login com o seu usuário comum, não como root! NUNCA como ROOT no ambiente gráfico!

```
# /etc/init.d/kdm start
```

2) Ainda como root, instale o pacote xnest, Nested X server:

```
# aptitude install xnest
```

Agora, em uma shell dentro do ambiente gráfico, vamos executar o Xnest, desabilitando qualquer controle de acesso:

3) (Abra um xterm e execute)

```
$ Xnest :1 -ac &
```

4) Faça uma conexão ssh no servidor remoto com as aplicações:

```
ssh <seu_usuario>@<ip_servidor>:1
```

5) Já no servidor, exporte a variável DISPLAY para a sua máquina na screen 1, ou seja, onde o Xnest está ``escutando'':

```
# export DISPLAY=<IP>:1
```

6) Agora já podemos testar uma aplicação gráfica, que irá rodar dentro da tela do Xnest:

```
# wmaker &
```

Como estamos acessando a aplicação remotamente, os processos da aplicação estão consumindo recursos do servidor, e não na minha máquina.

16.9.5. Servidor X Remoto

O Xterminal é um recurso dos servidores gráficos X presentes em todos os servidores Linux. Este recurso possibilita que uma máquina com menor desempenho possa executar uma aplicação gráfica a partir de um servidor, onde toda a carga de processamento está sendo realizada nele, e a nossa estação atuando somente como um terminal.

O Xterminal utiliza o protocolo XDMCP.

Utilizaremos o display manager gdm para fazer esse serviço. Vamos editar o arquivo onde ativaremos o XDMCP para o gdm.

vi /etc/gdm/gdm.conf

1) Localize o bloco [Xdmcp], utilizado para configuração desse protocolo. Ao encontrar esse bloco, ative o XDMCP alterando de **Enable=false para Enable=True:**

```
 [xdmcp]
Enable=true
```

Pronto! Basta reiniciar o gdm que ele já estará ``escutando'' na poa a 177/tcp.

2) Reinicie o gdm:

```
 # /etc/init.d/gdm stop
# /etc/init.d/gdm start
```

3) Verifique que a port 177/ucp está aberta:

```
 # netstat -nltup |grep 177
# fuser 177/ucp
```

4) Em sua máquina, faça uma requisição XDMCP à máquina de um dos colegas, mandando utilizar a screen 1:

```
 # X -query <IP> :1
```

16.10. Exercícios

1) Qual é a variável que define para onde vão as saídas gráficas? Como você define que a saída gráfica irá para a máquina ip 192.168.200.200, na screen 1 e display 0?

2) Qual é o arquivo que eu defino qual o meu padrão de inicialização, e qual a linha que eu defino?

3) Qual arquivo eu armazeno qual Display Manager estou utilizando? Como faria para trocar?

4) Qual é o caminho do executável do KDE e do Gnome?

5) Qual comando eu consigo utilizar, para configurar o daemon do XDM?

6) Qual é o arquivo de configuração do GDM, e do servidor Xorg?

7) Quais såo os Windows Managers que você conhece?

Capítulo 17

Instalação Linux em Desktop

17.1. Objetivos

Neste capítulo iremos aprender a fazer uma instalação básica em desktop de um sistema Debian GNU/Linux.

17.2. Instalando o Debian 4.0 - Etch

Por incrível que pareça, um desktop dá tanto trabalho quanto um servidor, pois será nesse ambiente que você irá trabalhar com mais ferramentas. Para instalarmos um Linux em um desktop precisamos saber para qual finalidade precisamos do sistema.

17.3. Perfil da instalação:

- **Partições -** O esquema de particionamento que adotaremos para nosso desktop é necessário por uma questão de infra-estrutura do laboratório didático. Este esquema deverá ser alterado e personalizado conforme cada caso.
- **Pacotes -** Os pacotes de um desktop são muito relativos, mas como iremos usar um perfil de desktop doméstico, partiremos do princípio de que teremos um usuário avançado de informática, com acesso à banda larga e que necessita de pacotes multimídia.

- **Usuários -** Por segurança, você irá definir um usuário padrão, cujo login será aluno e a senha 123456.
- **Pós-Instalação -** Para um desktop, o procedimento de pós-instalação é: atualização do sistema.

17.3.1. Telas de Instalação

Bootando pelo CD

Insira o CD no drive e reinicie seu computador. Logo, aparecerá a tela de boot. Não tecle enter! Siga as orientações seguintes passo a passo.

1) Abaixo, está mostrada a primeira tela que aparecerá após o boot da máquina. Não utilizaremos a opção padrão de instalação, mas uma diferente.

Ilustração 19: Tela inicial

2) Para sabermos quais opções existem, selecione help;Acima você pode navegar pelas telas de ajuda.

```
                    Welcome to Debian GNU/Linux!                          F1

This is a Debian 5.0 (lenny) installation CD-ROM.
It was built 20090413-00:10; d-i 20090123lenny1.

HELP INDEX

KEY     TOPIC

<F1>    This page, the help index.
<F2>    Prerequisites for installing Debian.
<F3>    Boot methods for special ways of using this CD-ROM
<F4>    Additional boot methods; rescue mode.
<F5>    Special boot parameters, overview.
<F6>    Special boot parameters for special machines.
<F7>    Special boot parameters for selected disk controllers.
<F8>    Special boot parameters for the install system.
<F9>    How to get help.
<F10>   Copyrights and warranties.


Press F2 through F10 for details, or ENTER to boot: _
```

Ilustração 20: Help

3) Selecione Graphical Install para iniciarmos nossa instalação no modo

Ilustração 21: Instalação Gráfica

gráfico.

4) Logo após o caregamento da interface gráfica, a primeira tela de instalação é sobre o Locale.
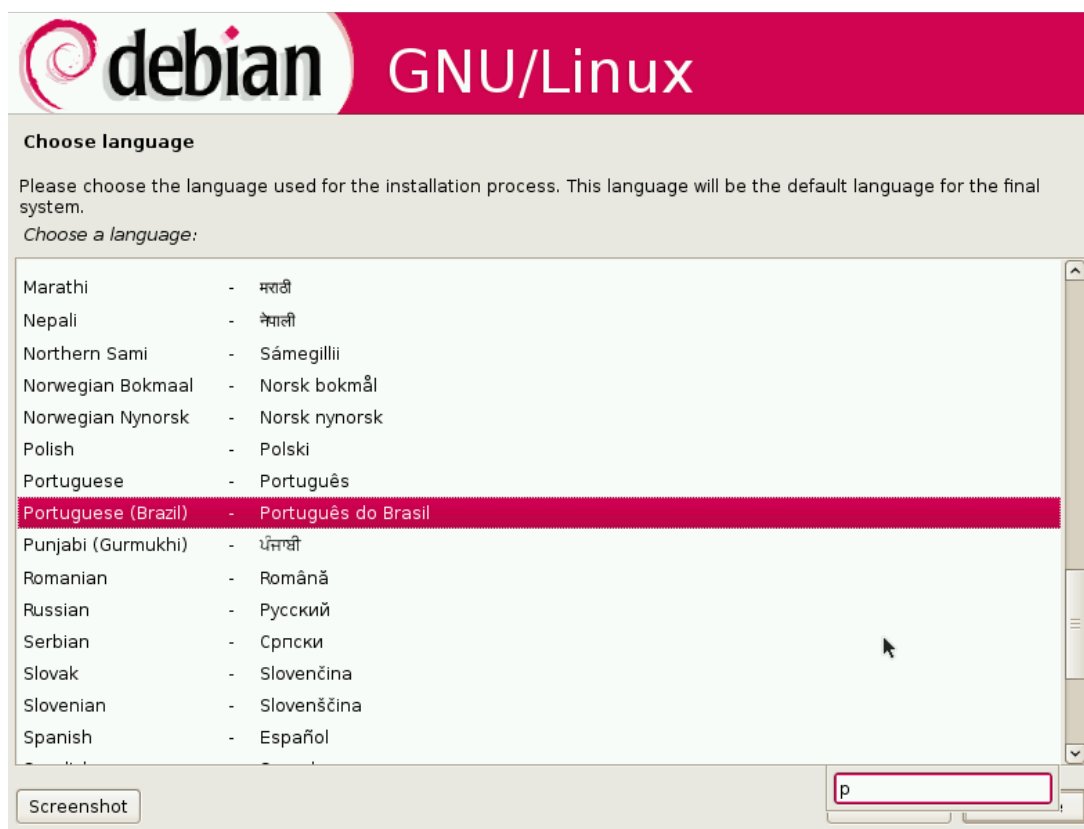
Ilustração 22: Locale

5) Selecione seu Pais:

Ilustração 23: Pais

6) O próximo passo é escolher o layout do teclado:

Preste atenção nesse ponto, pois muitos de nós ainda usamos teclados do layout US international

Ilustração 24: Layout Teclado

7) Deixe que o sistema detecte o dispositivo de media:

Ilustração 25: Carregando drom

8) Após carregar os componentes o sistema automaticamente carrega os componentes necessários para a instalação.
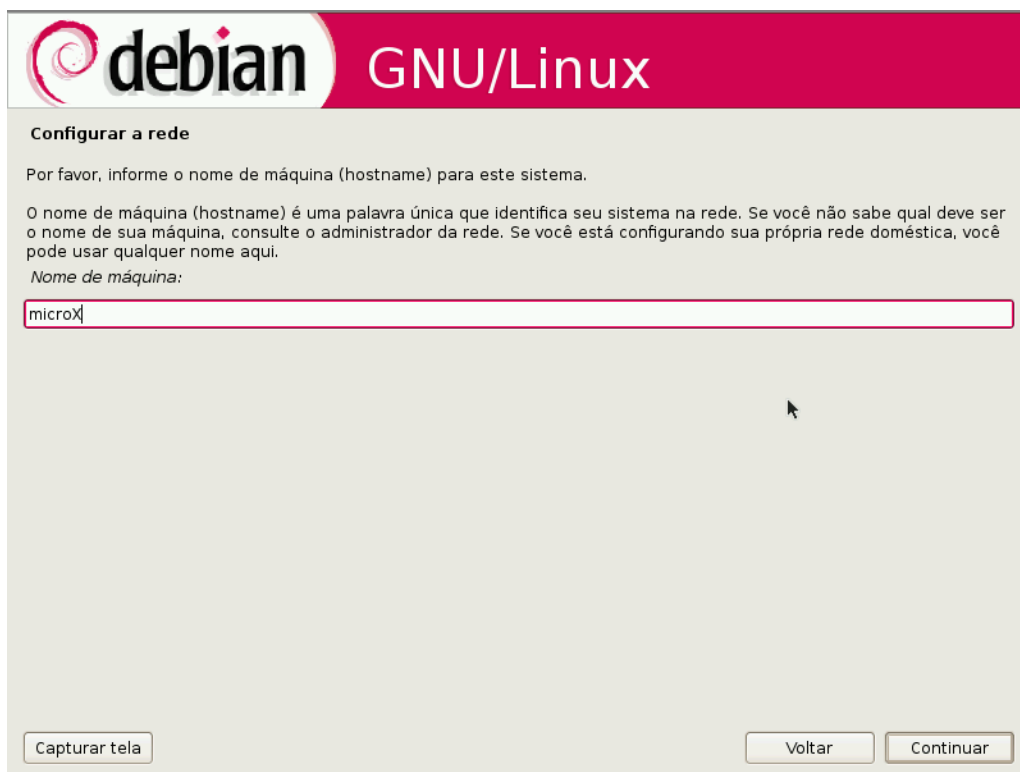
Ilustração 26: Carregando módulos da instalação

9) Após o carregamento dos módulos necessários o sistema irá configurar a rede através do protocolo dhcp.

Ilustração 27: Configuração automática de rede

10) O próximo passo agora é inserir o nome da máquina.

Ilustração 28: Hostname

11) Logo após entre com o domínio:

Ilustração 29: Domain Name

12) Selecione seu estado para que o sistema busque o melhor servidor NTP para sincronizar:
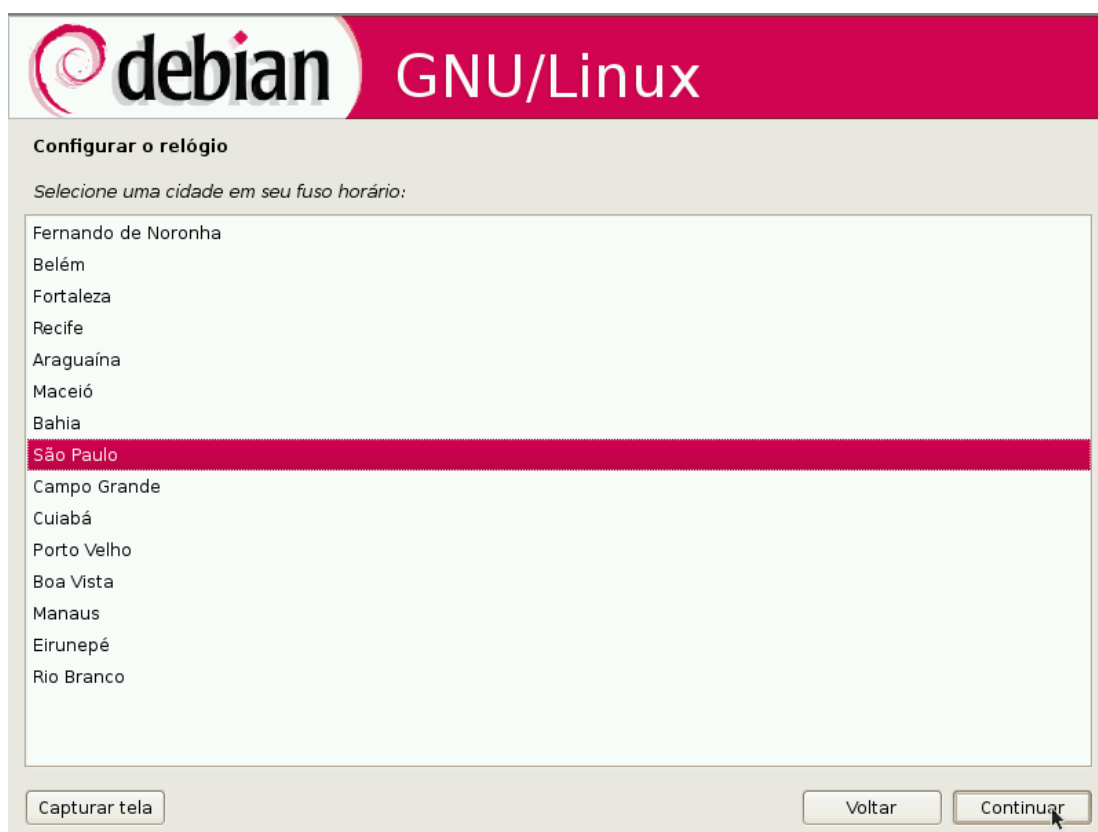
Ilustração 30: Timezone - NTP

13) Na próxima etapa da nossa instalação temos que particionar nosso disco fique atento na nossa tabela de partição:

Número	***Tipo***	***Ponto de Montagem***	***Tamanho***	***Sistema de Arquivos***
1	Primária	/boot	200MB	ext3
2	Primária	/	2GB	Ext3
3	Primária	/home	5GB	Ext3
5	Lógica	/usr	10GB	Ext3
6	Lógica	/var	256MB	Ext3
7	Lógica	/tmp	256MB	Ext3
8	Lógica	/var/log	3GB	Ext3
9	Lógica	swap	512MB	swap
10	Lógica		512MB	Não Utilizar
11	Lógica		512MB	Não Utilizar
12	Lógica		512MB	Não Utilizar
13	Lógica		512MB	Não Utilizar
14	Lógica		512MB	Não Utilizar

14) Selecione particionamento manual:

Ilustração 31: Particionamento Manual

15) Para zerar nossa tabela de particionamento, basta selecionar nosso disco em questão e confirmar a remoção da tabela de particionamento antiga.
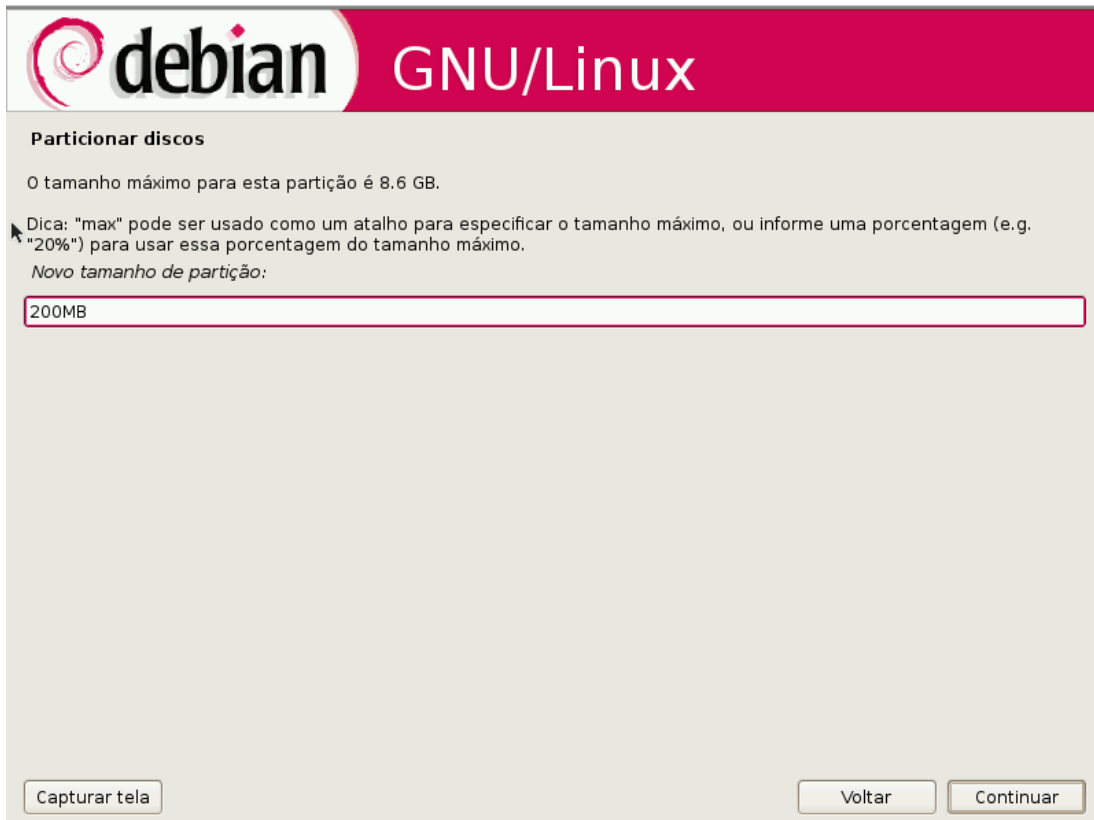
Ilustração 32: Tabela de particionamento

Ilustração 33: Nova Partição

Agora com a nossa tabela de partições limpa, sempre que quisermos adicionar uma partição devemos selecionar o ESPAÇO LIVRE e seguir os seguintes passos:

16) Indicar o espaço a ser usado:

Ilustração 34: Tamanho

17) Escolher entre primária ou lógica para a nova partição a ser criada:

Ilustração 35: Primária ou Lógica

18) Selecione agora se a partição nova ficará no inicio ou no fim do disco:

Ilustração 36: Inicio ou fim

19) Após seguir todas essas etapas, a tela abaixo sobre configurações da nova partições irá aparecer
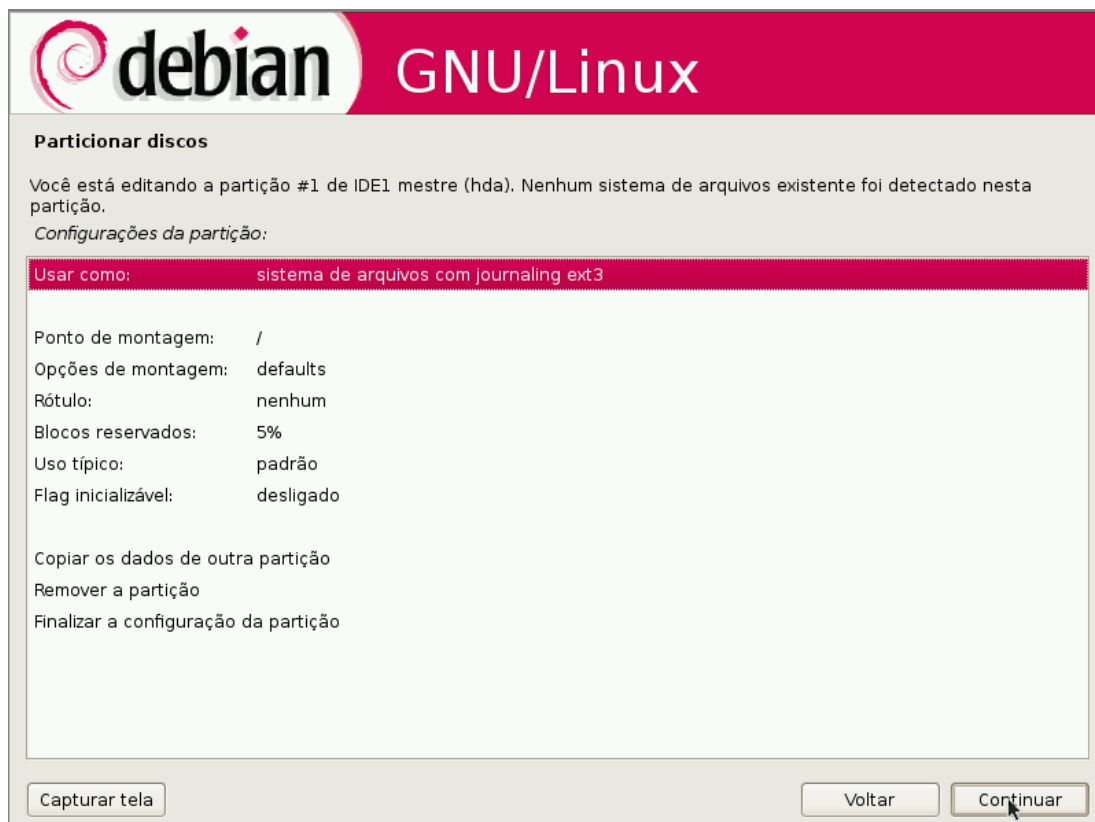
Ilustração 37: Configurando a nova partição

20) Dentro dessas configurações selecione o sistema de arquivos a ser usado:

Ilustração 38: Sistema de arquivos

21) Logo após selecione o ponto de montagem:
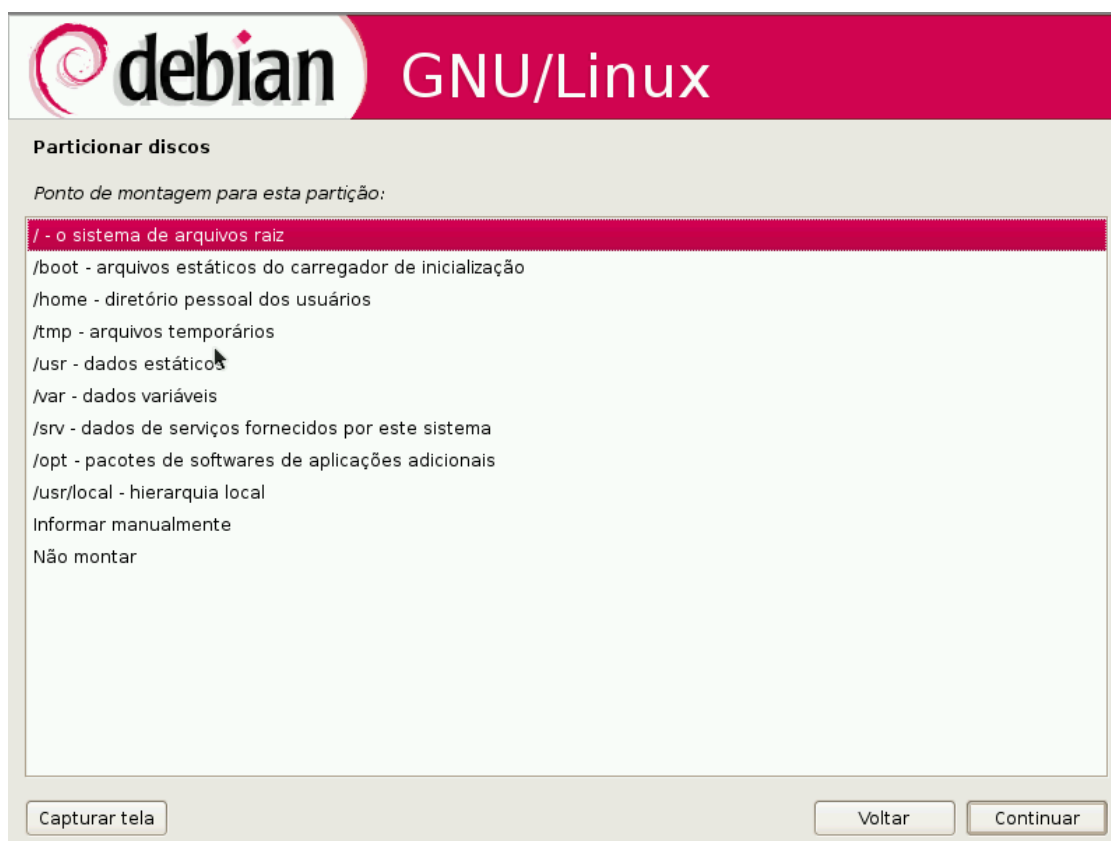
Ilustração 39: Ponto de montagem

22) Configure agora as opções do FSTAB:
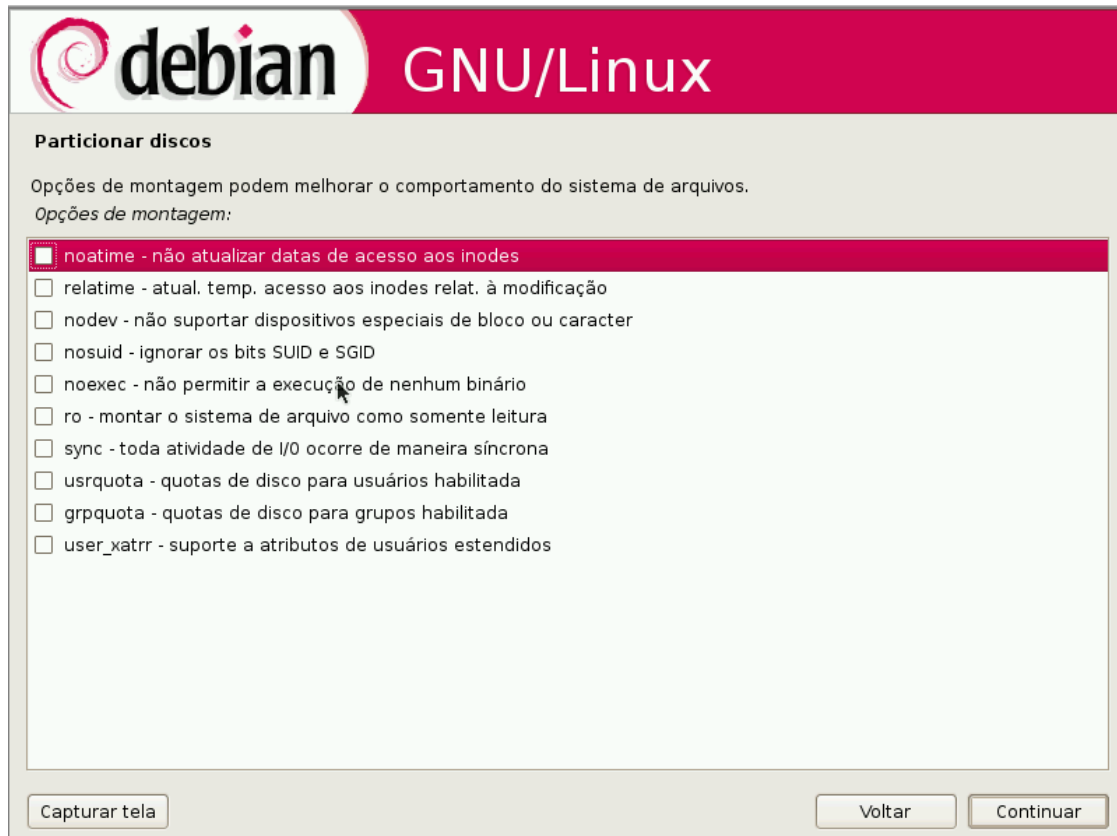
Ilustração 40: Opções do FSTAB

23) Ao final da configuração de uma partição você pode acompanhar as partições que já foram criadas.
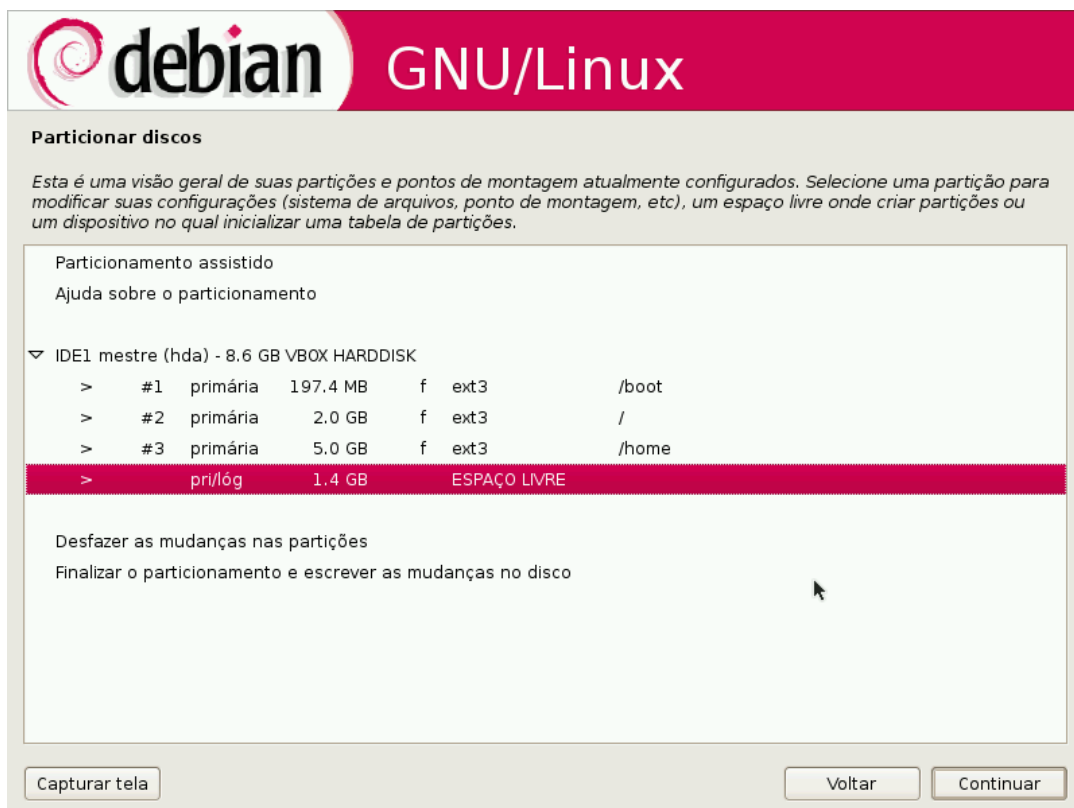
Ilustração 41: Particionador

24) Nossa próxima etapa é finalizar o particionamento e pedir para que o sistema grave essas mudanças no disco.

Ilustração 42: Gravar mudanças no disco

25) Logo após o particionamento da máquina, o sistema base será automaticamente instalado.

Ilustração 43: Instalação do Sistema Base

26) Após a instalação do sistema base é hora de configurar as credenciais do sistema, insira a senhas para o administrador root.

Ilustração 44: Senha do admin

27) A próxima etapa é criar um usuário comum para que não utilizarmos o usuário root de forma indevida.

Ilustração 45: Nome completo

Ilustração 46: Nome de usuário

28) Determine a nova senha para esse usuário:

Ilustração 47: Senha do usuário comum

29) Na próxima etapa da instalação você tem a possibilidade de catalogar mais fontes de media para obter uma instalação mais farta.

Ilustração 48: Novas medias

30) Chegou a hora de configurar os espelhos de rede. Eles são servidores externos da onde o instalador pode baixar pacotes para completar a instalação.

Ilustração 49: Configurar gerenciados de pacotes

31) Após as configurações de pacotes o instalador irá configurar a ferramenta aptitude para fazer a instalação desejada.

Ilustração 50: Configurando gerenciador

32) O instalador irá te perguntar sobre o Popularity-contest, que é a ferramente que determina quais serão os pacotes do primeiro CD nas próximas versões:
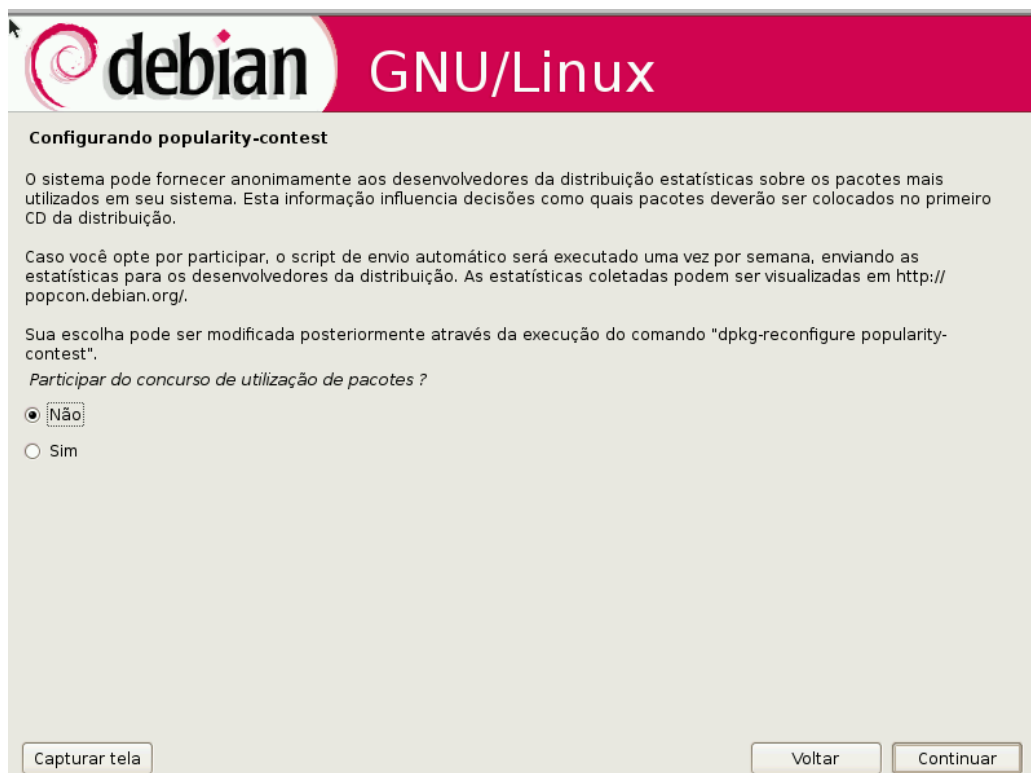
Ilustração 51: Concurso de Popularidade

33) Enfim o programa tasksel irá lhe oferecer os tipos de instalação disponiveis:
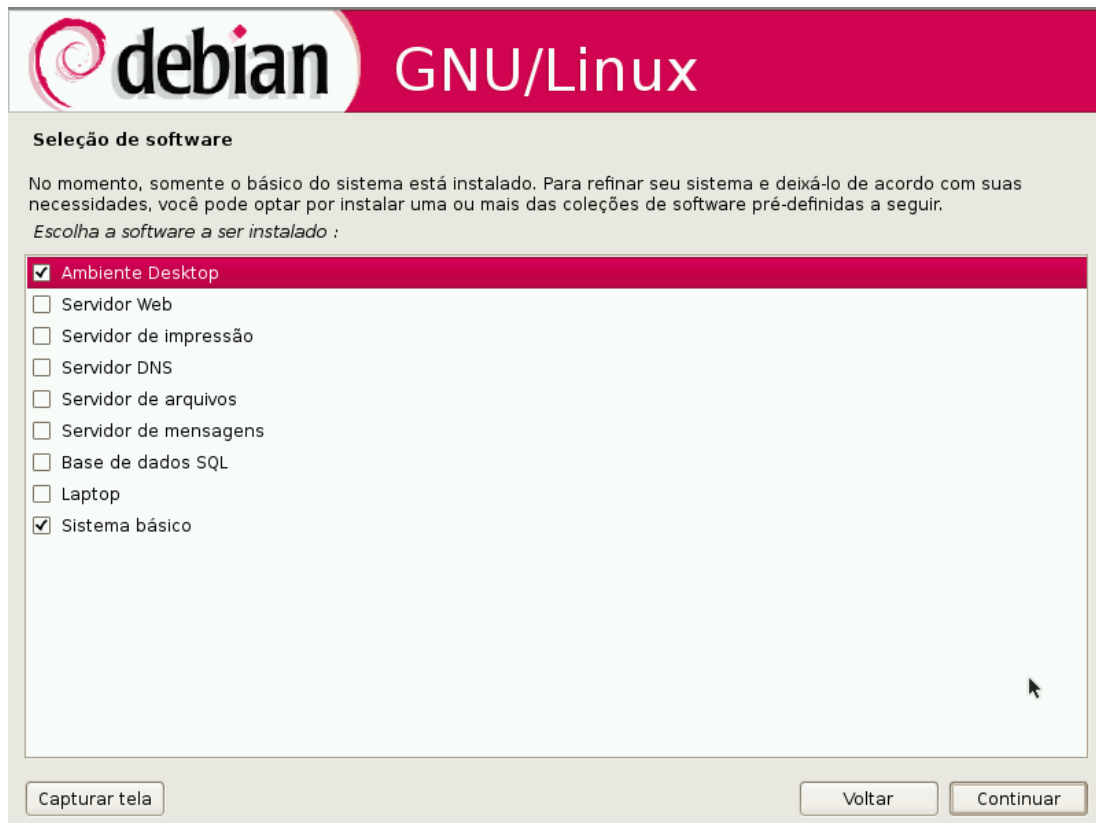
Ilustração 52: Tasksel

34) Após selecionar o sistema irá baixar os pacotes necessários:
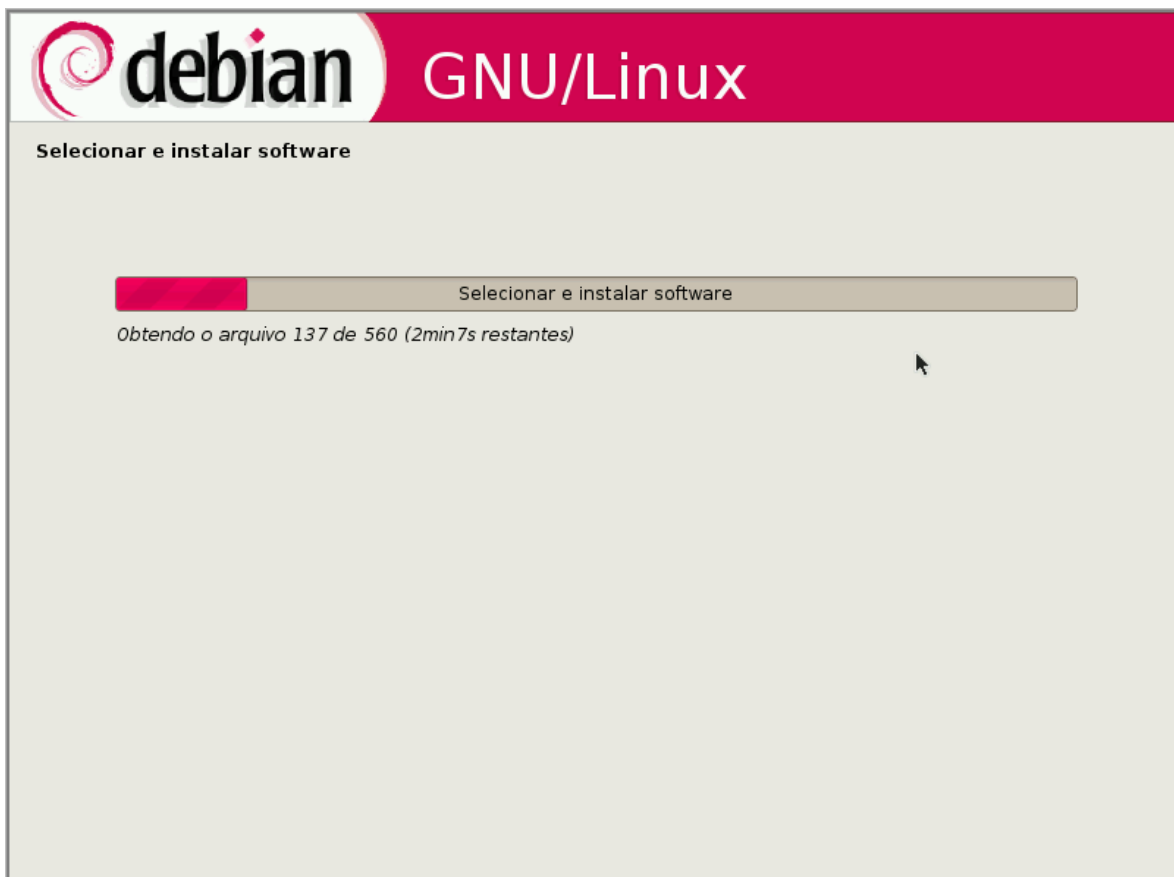
Ilustração 53: Selecionando pacotes

35) Logo após irá começar a instalação em massa dos softwares:

Ilustração 54: Instalando pacotes

36) Quando a instalação acabar você deve instalar o GRUB na sua MBR:

Ilustração 55: Instalação do Grub

37) Parabéns por sua nova instalação!!! Agora é só tirar o CD do driver e rebotar a máquina.

Ilustração 56: Retirando a media

38) Bem vindo ao seu novo sistema:
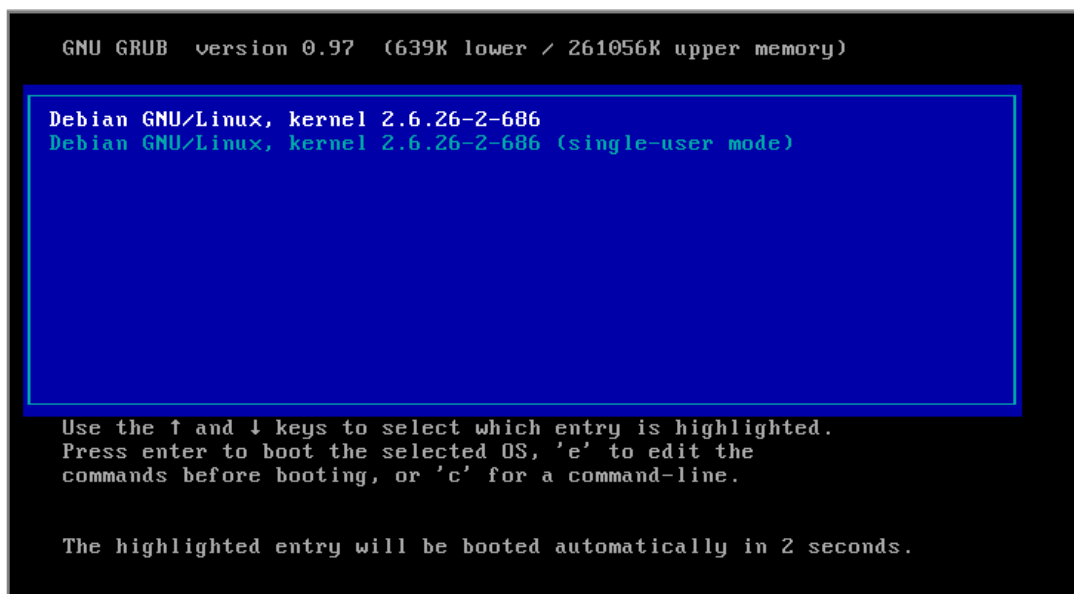
Ilustração 57: novo sistema

ANEXOS

System Imager - 4Linux

O que é

O System Imager é um sistema de automatização para rotinas de backup e recuperação de máquinas. Ele permite que as mesmas máquinas possam ser compartilhadas por vários cursos simultâneos, mas de forma que o estado delas, isto é, todos os seus arquivos e configurações sejam guardados e recuperados de forma individual por aluno.

Instalando o programa.

Primeiramente, o aluno deverá fazer o download do programa e mudar as suas permissões. Esse procedimento só é necessário na primeira aula.

```
 # cd /sbin
# wget 192.168.1.1/si/si_cliente
# chmod u+x /sbin/si_cliente
```

Utilizando o System Imager

17.3.2. Backup ao final de cada aula

Ao **final** de cada aula o aluno executa o comando abaixo, selecionando a opção de **Enviar Imagem.**

```
# si_cliente
```

O professor verificará se todas as maquinas estão com a imagem pronta para enviar. Em caso positivo, irá executar um programa para recebê-las.

PORTANTO: NÃO desligue seu micro, pois o servidor estará conectado a ele recebendo os arquivos modificados. Após o processo ter sido concluído, as máquinas serão desligadas automaticamente.

17.3.3. Restore antes de cada aula

No início de cada aula, a imagem de cada máquina deverá ser restaurada. Para isso, basta executar o comando abaixo, selecionando a opção Receber Imagem. Normalmente, esta operação é realizada pelo próprio instrutor antes da aula se iniciar e deverá ser realizada pelo aluno apenas sob sua orientação.

ATENÇÃO! Todos os arquivos do sistema poderão ser apagados! Se você não fez nenhum tipo de backup, faça-o antes!

```
# si_cliente
```

Aparecerá uma listagem das imagens disponíveis. Você deverá escolher aquela que corresponder a sua máquina.

Um exemplo de nomenclatura das imagens:

```
 450-Instrutor-31-Noturno-10
(Cod. do curso)-(Nome do Instrutor)-(Dia de Início)-(Período)-(Fim do
IP do Micro)
```

Após a conclusão do processo, a máquina irá se reinicializar automaticamente e, em seguida, estará pronta para uso.

Manipulando Hardware e Dispositivos

Objetivos

Neste capítulo iremos aprender de que forma os dispositivos de hardware são mapeados e manipulados no Linux. Para que esse assunto faça mais sentido, primeiramente veremos alguns conceitos sobre arquitetura de computadores e dispositivos de hardware.

Arquitetura de Computadores e Dispositivos de Hardware

Podemos dividir um computador em 3 partes principais: CPU, memória RAM e dispositivos. A CPU, muitas vezes denominada como o cérebro do computador, é responsável por executar todo o processamento das informações, que são armazenadas na memória RAM.

Mas, um computador não tem muita utilidade se não for capaz de se comunicar com o mundo exterior. Um teclado e um monitor, ou uma rede, são exemplos de meios de comunicação. Até mesmo um simples botão (no lugar do teclado) e uma lâmpada (no lugar do monitor) poderíamos considerar como exemplo. A esses elementos damos o nome de dispositivos de hardware, e incluem interfaces de rede, controladoras de disco, as próprias unidades de disco, portas seriais, paralelas e USB, apenas para exemplificar.

Arquitetura do computador é o nome que damos à forma como essas 3 coisas

são organizadas numa máquina. A figura a seguir ilustra a arquitetura típica dos PCs.
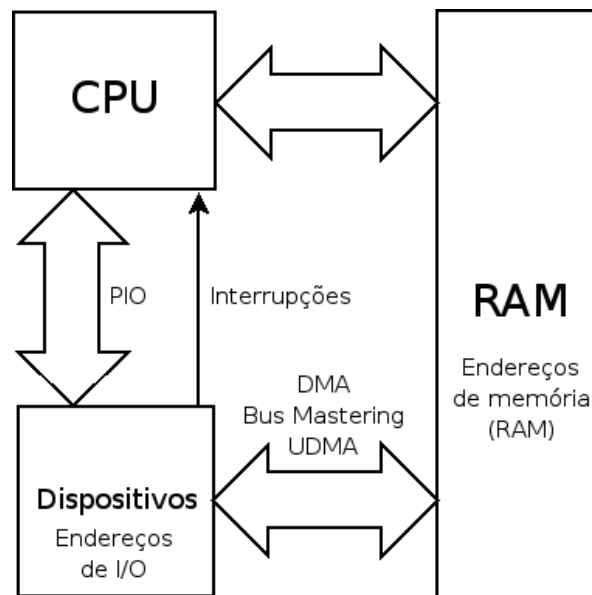
Ilustração 58: Arquitetura

Acesso aos dispositivos

O acesso aos dados da memória RAM é feito de forma rápida e eficiente através de otimizados canais de comunicação. Entretanto, o acesso aos dispositivos é mais lento, e as tecnologias responsáveis por essa função podem ser dividas em duas categorias.

A primeira, chamada PIO (Programmed Input/Output), envolve a CPU na transferência das informações. Para identificar os dispositivos, são associados a eles os chamados endereços de I/O. Assim, por exemplo, a COM1 tem o endereço 3F8h, a LPT1 o endereço 378h. Na verdade, um certo intervalo desses endereços são utilizados para cada dispositivo. Esses endereços podem ser consultados no arquivo /proc/ioports.

Além desses endereços, em alguns casos temos um interrupção associada a um dispositivo. Isso porque, como são mais lentos que a CPU, precisam de algum mecanismo para informar à CPU de que o trabalho terminou. Do contrário, a CPU

teria de ficar constantemente consultando o dispositivo para saber quando enviar ou ler o próximo byte, e conseqüentemente perdendo tempo.

A cada dispositivo, é associada uma interrupção. Entretanto, o número disponível de interrupções é limitado, e por essa razão, pode faltar alguma e/ou ocorrer os famosos conflito de interrupção. As interrupções utilizadas podem ser consultadas no arquivo /proc/interrupts.

Entretanto, a tecnologia PIO limita a velocidade de transferência de dados. Ela é apropriada apenas para dispositivos como teclado, portas seriais e paralelas, unidades antigas de CD-ROM, etc.

Outro problema relacionado a ela é o envolvimento da CPU. Isso porque, vários ciclos de processamento são perdidos no processo de transferência dos dados, o que se agrava tanto quanto maior for a velocidade dessa transferência.

Para contornar essa situação, foi criado o DMA (Direct Memory Access). Essa tecnologia permite que o dispositivo acesse diretamente a memória RAM, escrevendo ou lendo dados, sem interferência da CPU. Para isso, são utilizados os chamados canais de DMA, um para cada dispositivo e também uma controladora de DMA. Os canais utilizados podem ser consultados no arquivo /proc/dma.

Mas essa tecnologia, desenvolvida para os antigos barramentos ISA, também ficou ultrapassada, e cedeu lugar ao Bus Mastering. Nesse caso, o próprio dispositivo faz todo o controle de acesso a memória RAM, de modo que os canais de DMA não são mais necessários. Essa nova tecnologia permitiu o surgimento do UDMA (Ultra DMA).

Embora caindo em desuso atualmente, alguns dispositivos legados possuem endereços e interrupções

padrões. A LPI costuma cobrar essas informações em suas provas. Memorize a tabela abaixo antes da prova!</lpi>

Dispositivos	Nome no Linux	End. Hex	Int.
COM1	/dev/ttyS0	3 F 8	4
COM2	/dev/ttyS1	2 F 8	3

COM3	/dev/ttyS2	3,00E+008	4
COM4	/dev/ttyS3	2,00E+008	3
LPT1	/dev/lp0	378	7
LPT2	/dev/lp1	278	5

Softwares Desktop

Objetivos

Existe hoje, uma infinidade de softwares aplicativos de alta qualidade para ambiente GNU/Linux. Temos disponíveis navegadores, leitores de e-mail, suítes office, programas para editoração de imagens e vídeo, modelagem 3D, simulações científicas, tocadores multimídia e vários outros. Neste capítulo, veremos apenas alguns exemplos dessas aplicações.

Ambientes Gráficos

Os ambientes gráficos em Linux são opcionais e podem ou não ser executados automaticamente. A escolha do ambiente gráfico mais adequado segue os mais diversos aspectos, como: consumo de recursos, performance, usabilidade e versatilidade. Um ambiente gráfico adequado para uma pessoa pode não ser adequado para outra. Contudo, a padronização em torno de um ambiente gráfico é fundamental em ambientes corporativos para minimizar os esforços da equipe de suporte com treinamento e para maximizar a base de conhecimento de problemas e soluções conhecidas. Os ambientes gráficos que veremos podem ser usados em qualquer sistema Linux. Apesar disso, algumas distribuições usam uma interface por padrão, mas nada o impede de usar uma interface diferente.

Vejamos algumas delas:
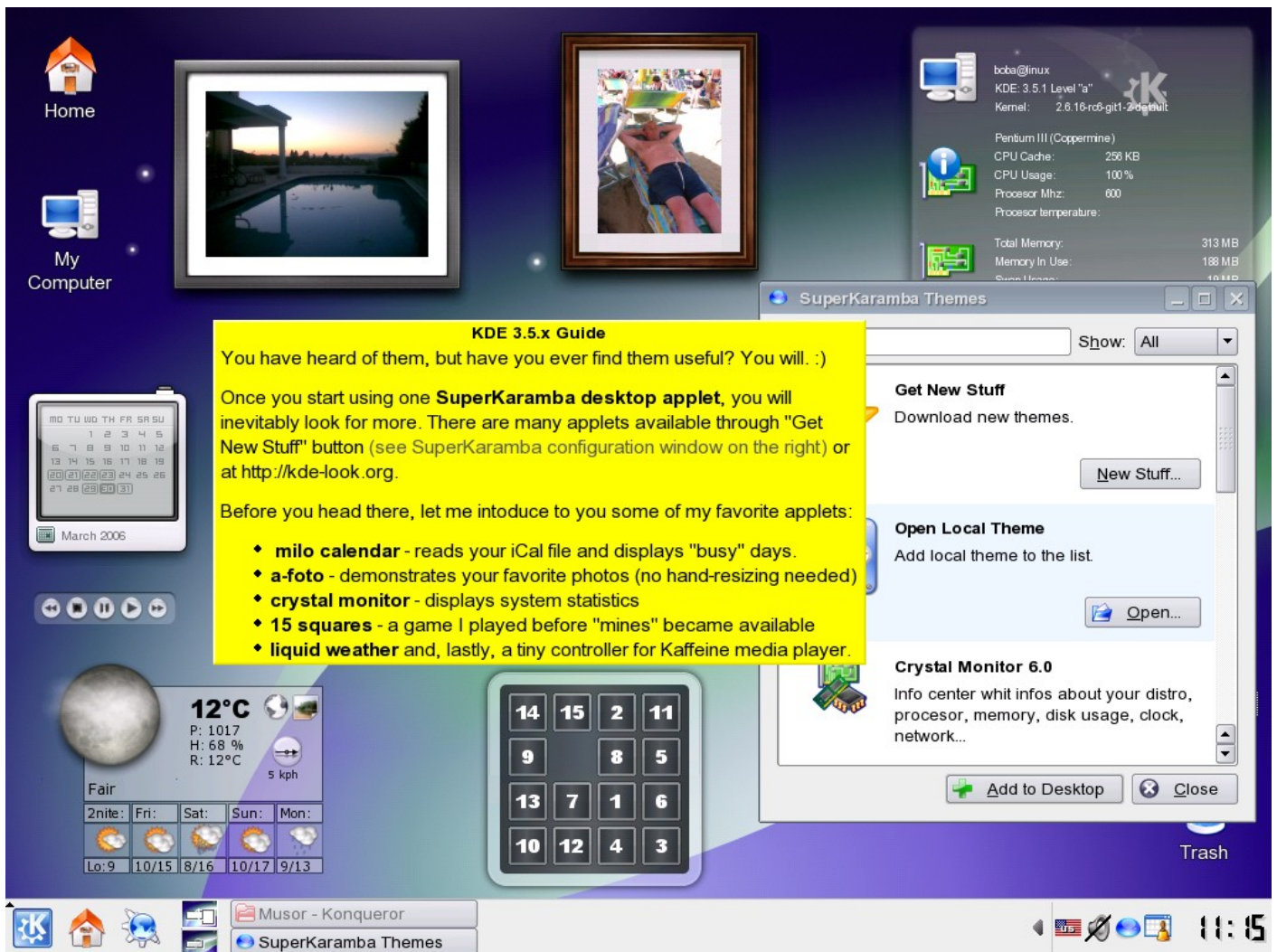
Ilustração 59: KDE
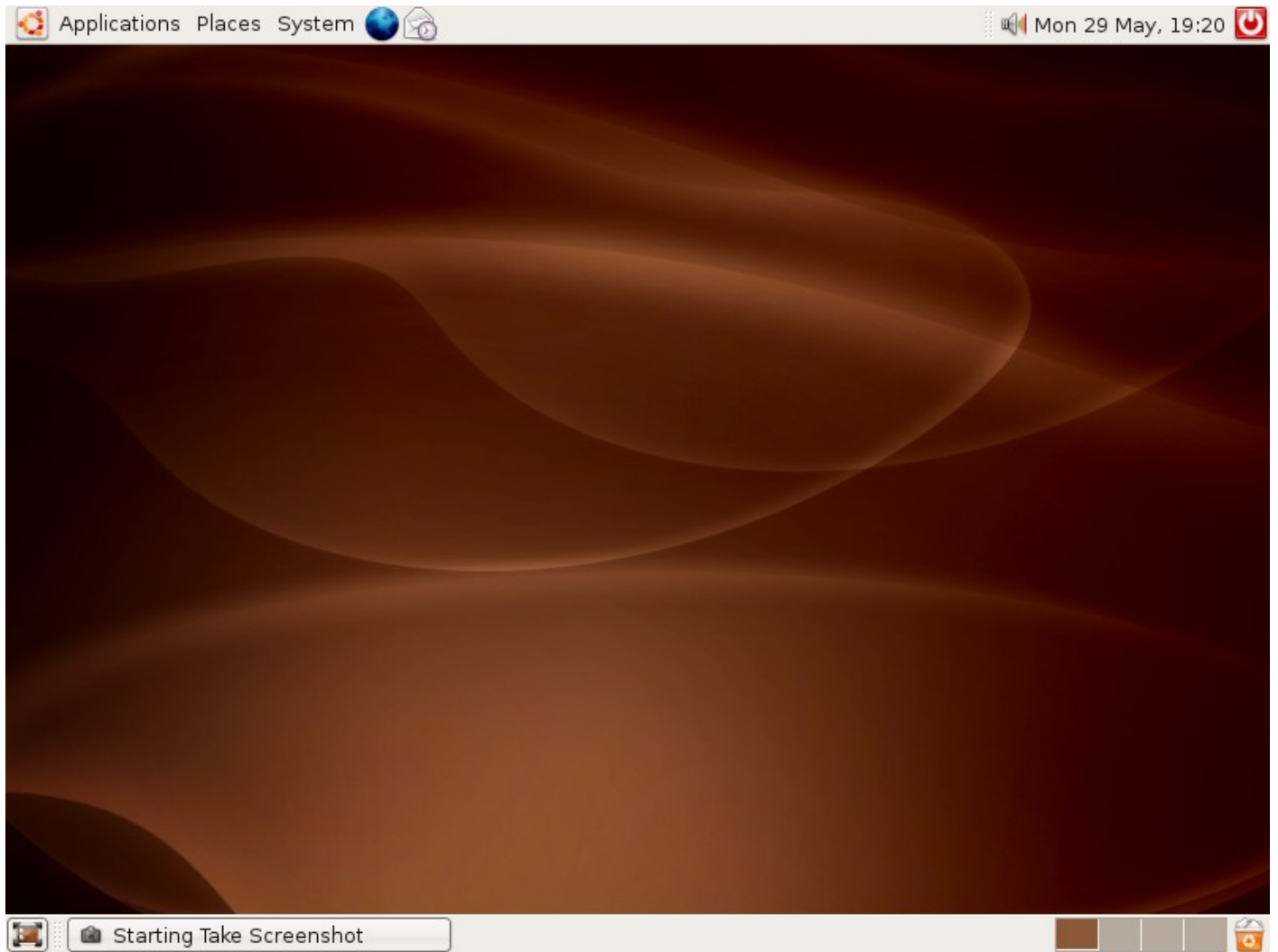
Ilustração 60: GNOME
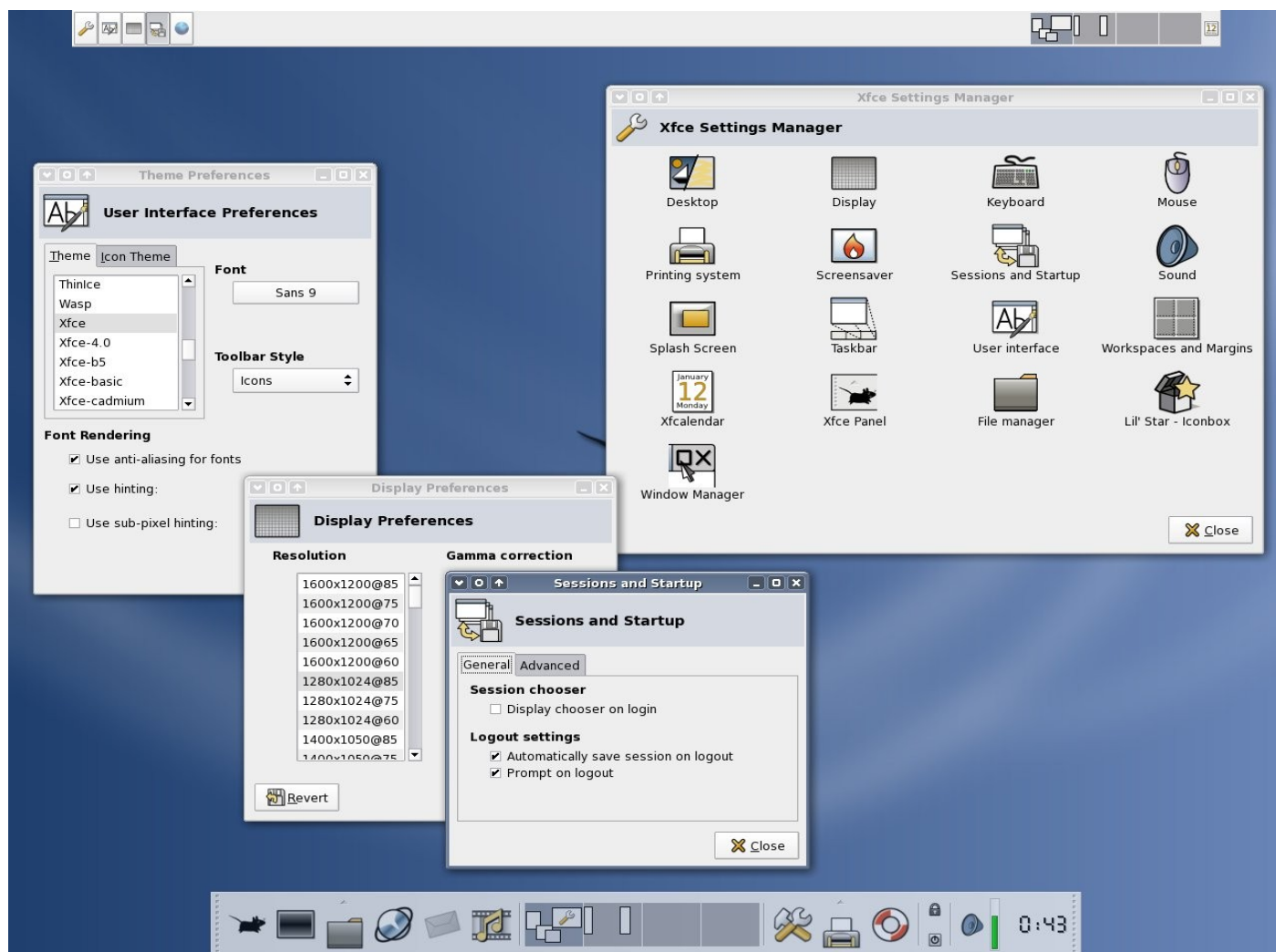
Ilustração 61: XFCE

Suites de Produtividade

Uma suite de produtividade é pré-requisito para a grande maioria dos computadores. Os usuários necessitam de ferramentas de edição de textos, planilhas eletrônicas e programas de apresentação para executar as suas tarefas diárias em ambientes corporativos.

Uma ferramenta domina o mercado de suítes de produtividade baseadas em Software Livre: o BrOffice.org, que pode ser obtido no site www.broffice.org.br. O projeto brasileiro traduz e mantém as versões em português do Brasil da suite de produtividade baseada no projeto mundial OpenOffice.org.

Editor de Textos
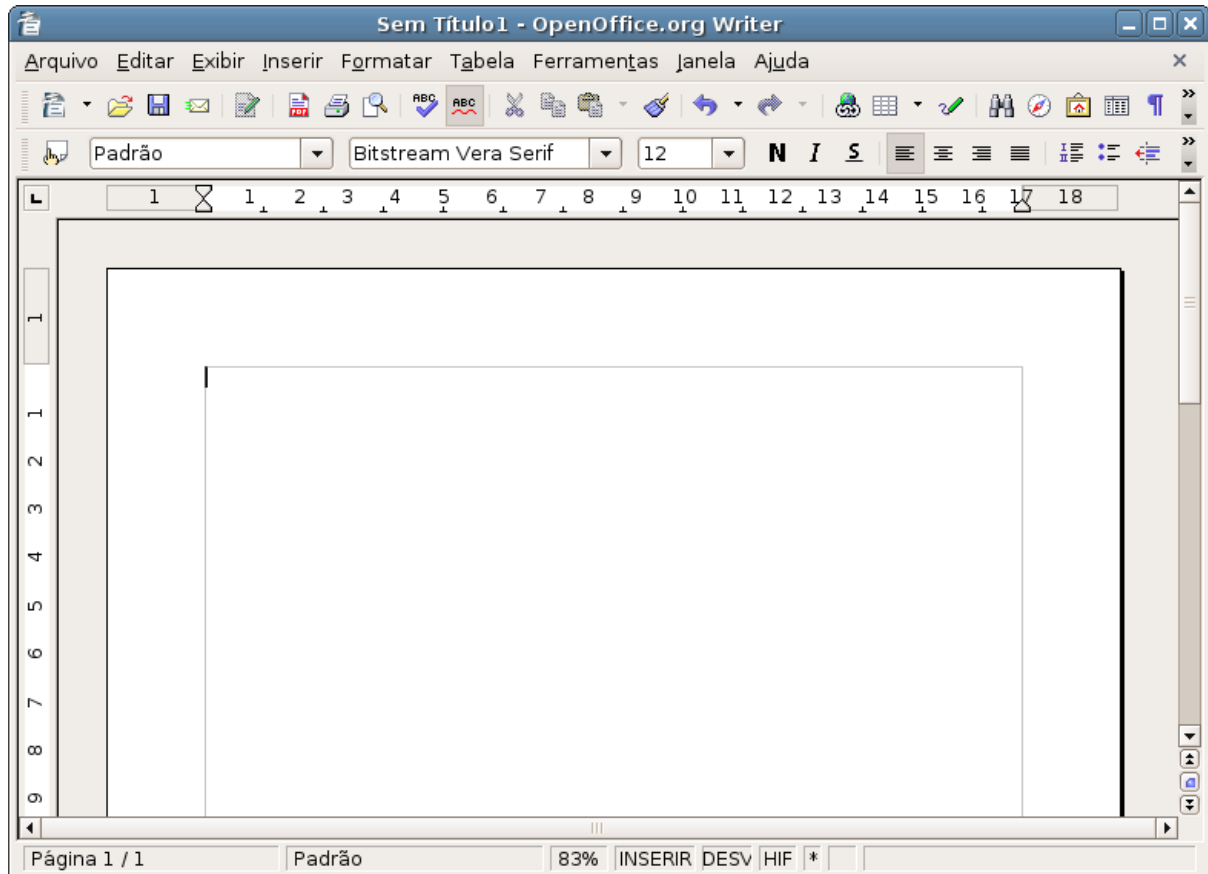
Ilustração 62: OpenOfice Writer

Planilha Eletrônica
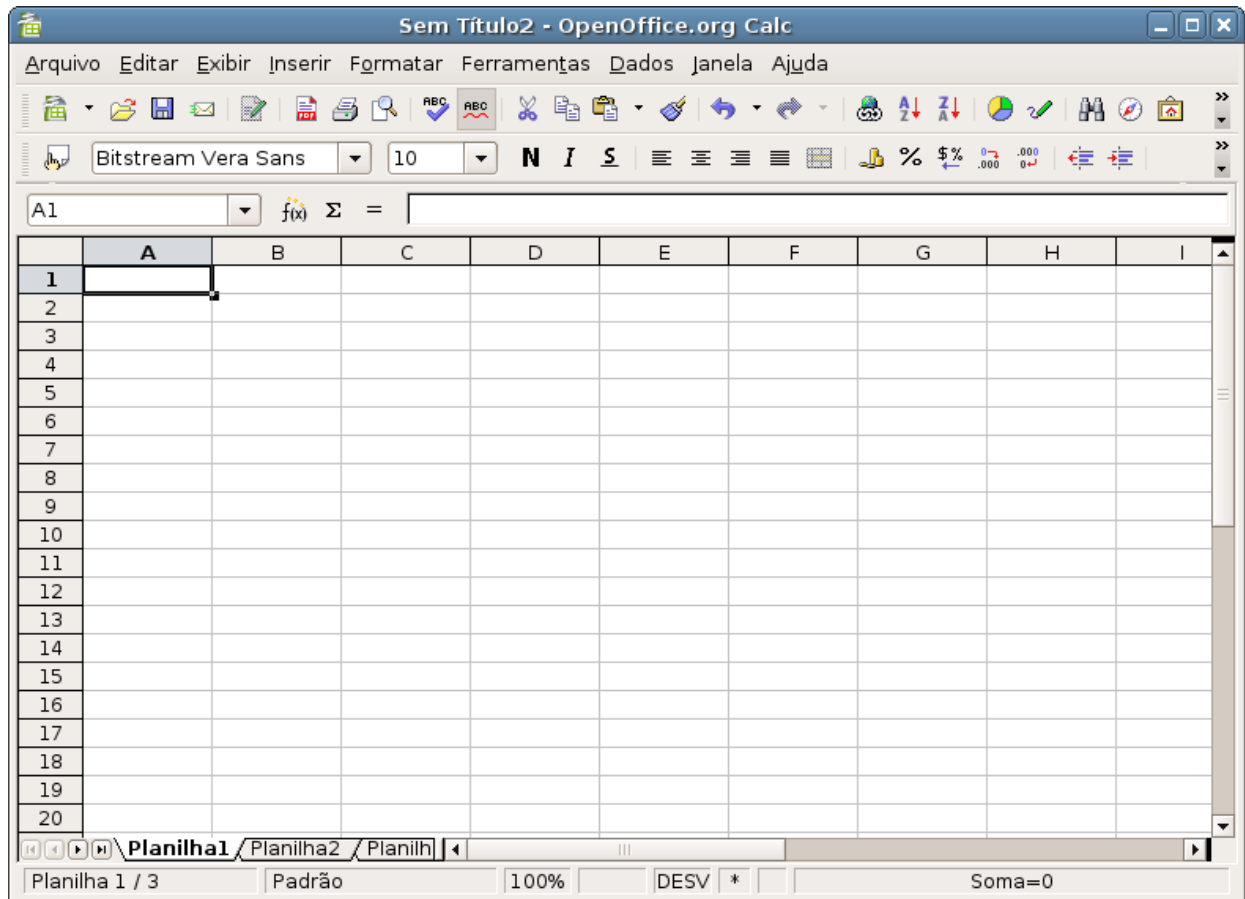
Ilustração 63: OpenOffice Calc

Programa de Apresentações
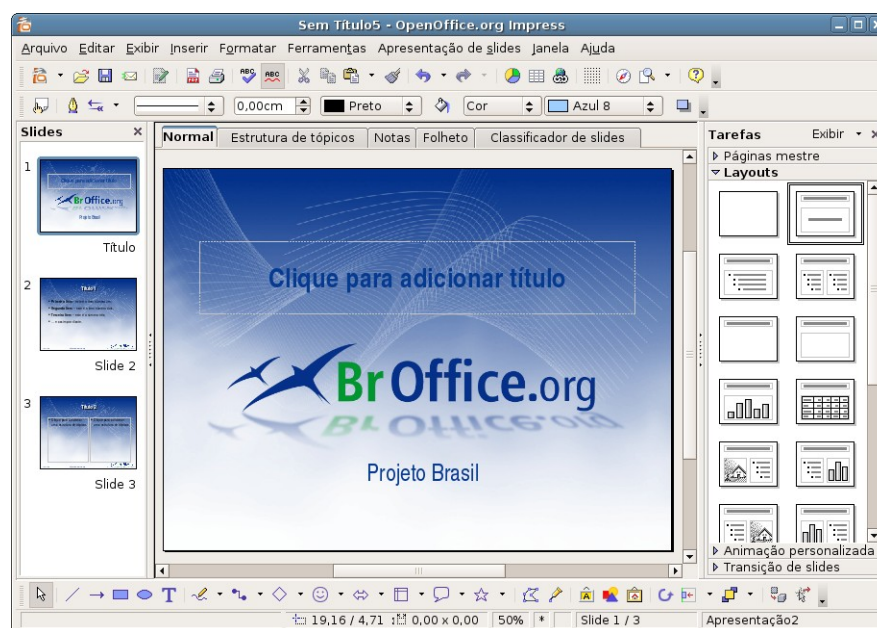
Ilustração 64: OpenOffice Impress

Internet

O navegador Firefox e o leitor de correio eletrônico Thunderbird são ferramentas avançadas cujos recursos são copiados por programas proprietários.

Disponíveis em diversas plataformas, representam hoje a maneira mais segura, rápida e fácil de navegação na Internet e gerenciamento de correio eletrônico pessoal.

Navegador
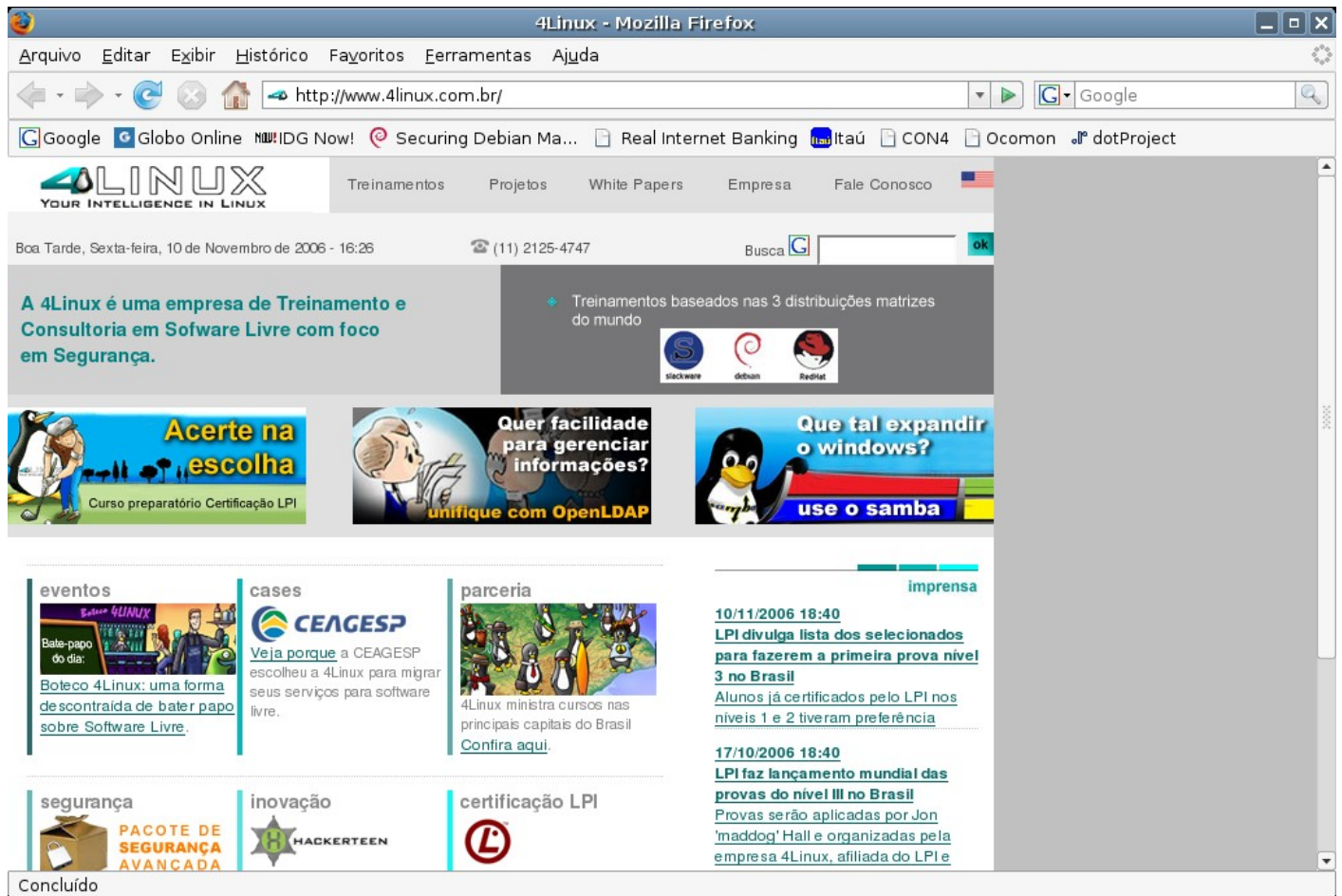
Ilustração 65: Firefox ou Iceweasel

Multimídia

Os programas gráficos em GNU/Linux têm recursos e usabilidades compatíveis com os mais modernos do mercado. Escutar músicas em mp3, assistir vídeos e muito mais são tarefas fáceis e intuitivas. A edição de fotos pode ser feita com bastante eficiência em programas avançados como o GIMP, um dos editores de fotos mais competentes do mercado.

Áudio

Ilustração 66: Xmms

Vídeo

Ilustração 67: Kaffeine

Gráficos

Ilustração 68: Gimp

Acessibilidade

A cada dia que se passa mais e mais pessoas vem precisando utilizar o computador. Muitas delas ainda tem uma grande dificuldade ou algum prolema físico envolvido. Os projetos se software livre sempre lembraram dessas pessoas como essas pessoas se lembraram do software livre. Hoje existem muitos casos de deficientes físicos ajudarem o software livre, pois foram ajudados a chegar na tecnologia através de softwares como Festival

Esse apêndice foi desenvolvido para apresentar alguns dos muitos módulos de acessibilidade do Linux. Para obtermos suporte ao modo accessibility no gnome como é chamado primeiro devemos instalar o pacote:

```
# aptitude install gnome-accessibility
```

Após essa etapa as ferramentas de acessibilidade foram disponibilizadas em Aplicações -> Acesso Universal.

Dasher

É um interface para escrita para pessoas que não podem usar o teclado. O teclado pode ser substituível por um joystick, uma leitor de pupila ou mesmo um mouse.
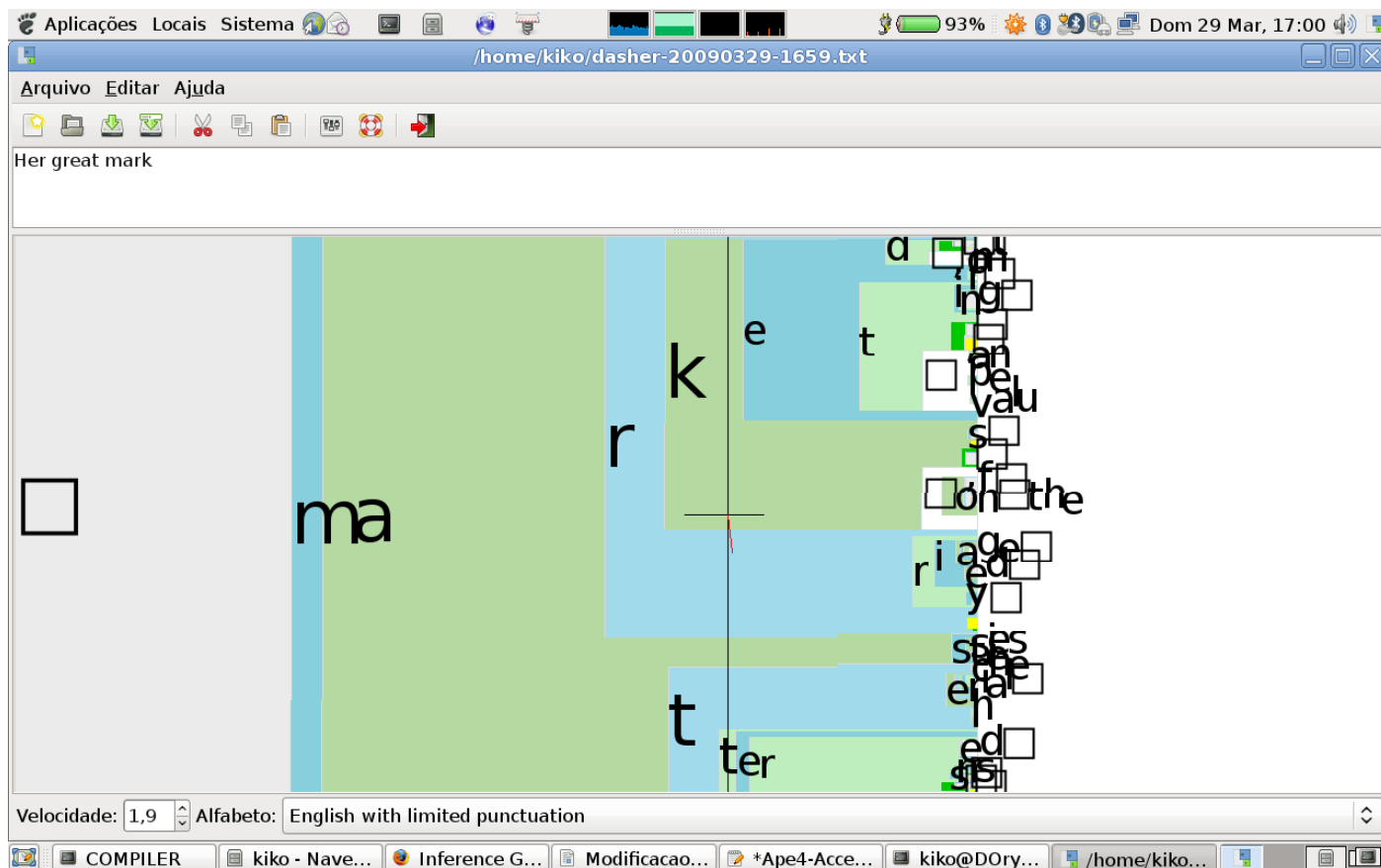
Ilustração 69: Dasher

GOK

O projeto GOK foi a selecionado dentre muitos outros projetos como o projeto que mais cumpre com a função de não se utilizar o teclado. Com ele é possível usar apenas o mouse e ou outros devices especiais.
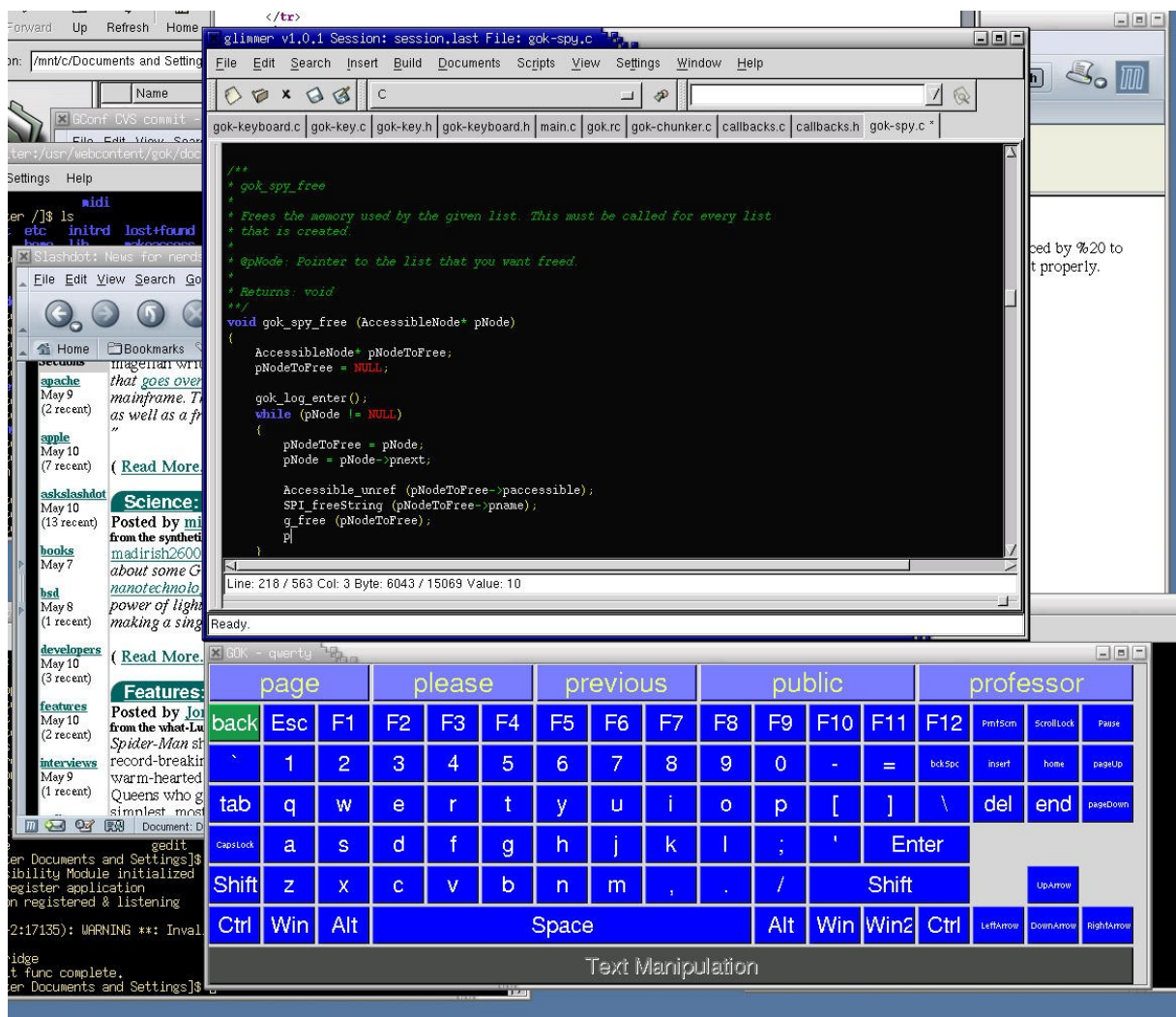
Ilustração 70: GOK

Festival

Software que pronuncia textos para linhas inseridas em uma STDIN.

Para nós administradores, podemos ver ele bem de baixo, sua API é bem parecida com python:

```
 Copyright (C) University of Edinburgh, 1996-2004. All rights reserved.
For details type `(festival_warranty)'
festival>
festival>
festival> (SayText "Hi Linux System")
```

REFERÊNCIAS BIBLIOGRÁFICAS

JULIO CESAR NEVES. *Programação em Shell Linux*. 6.ª edição. Brasport. 2006

The Linux documentation Project, website: http://www.tldp.org. Acesso em 28 de março de 2008.

Pritchard, Pessanha, Langfeldt, Stranger and Dean. Certificação Linux LPI 2.ª edição. AltaBooks. 2007

Gagné, Moving to Linux. 1ª edição, Addilson Wesley

Rubem E. Ferreira, Guia de Administração Linux. 2ª edição, Novatec, 2008